CSC-EPL-SUM-88/009

2

COMPUTER SECURITY CENTER

NATIONAL COMPUTER SECURITY CENTER

FINAL EVALUATION REPORT

OF

PRIME COMPUTER, INC.

PRIMOS
REVISION 21.0.1.DODC2A

18 July 1988

FINAL EVALUATION REPORT

PRIME COMPUTER, INC.

PRIMOS REVISION 21.0.1.DODC2A

NATIONAL
COMPUTER SECURITY CENTER

9800 Savage Road
Fort George G. Meade
Maryland 20755-6000

July 18, 1988

CSC-EPL-SUM-88/009
Library No. S231,362

FOREWORD

This publication, the Final Report for Prime Computer, Inc.'s PRIMOS Revision 21.0.1.DODC2A, is being issued by the National Computer Security Center under the authority of and in accordance with DoD Directive 5215.1, "Computer Security Evaluation Center". The purpose of this report is to document the results of the formal evaluation of PRIMOS Revision 21.0.1.DODC2A operating system. The requirements stated in this report are taken from DEPARTMENT OF DEFENSE TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA dated December 1985.

Approved:

[illegible] July 18, 1988

Eliot Sohmer
Chief, Office of Computer Security
Evaluations, Publications, and Support
National Computer Security Center

Accession For	
NTIS GRA&I	☑
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

ACKNOWLEDGEMENTS

Team Members

Team members included the following individuals, who were provided by the following organizations:

Richard B. Newton
David W. Summers
Harold J. Wolfe

MITRE Corporation
Bedford, MA 01730

Myron M. Coplin

National Computer Security Center
Ft. George G. Meade, MD 20755-6000

TABLE OFCONTENTS

FOREWORD iii

ACKNOWILEDGEMENTS iv

EXECUTIVE SUMMARY ix

Section 1 INTRODUCTION 1
- Evaluation Process Overview 1
- Document Organization 2

Section 2 SYSTEM OVERVIEW 3
- Background and History 3
- Hardware Architecture 4
- Major Hardware Components 5
  - CPU 5
  - Memory Management 12
  - Process Management 16
  - I/O Management 17
- Hardware Protection Mechanisms 19
  - Virtual Memory Segments 19
  - Hardware Protection Rings 20
  - Privileged CPU Instructions 23
- Hardware Diagnostics 23
- Software Architecture 24
- The PRIMOS Kernel 25
- Ring 0 System Processes 27
  - The Timer Server 27
  - The Copy Server 27
- Ring 3 Trusted Processes 28
  - The Login Server 28
  - The Audit Server 29
  - Distributed System Management (DSM) Servers 29
  - DSM Security Through Groups, Roles, and ACLs 30
  - Remote System User (RESUS) 30
  - System Information Metering (SIM) 30
  - Unsolicited Message Handler (UMH) 31
  - Hardware Event Logging 31
  - The Spooler Subsystem 31

The Batch Subsystem....31
Use of the Gate Mechanism....32
Parameter Checking....33
The Dynamic Linking Mechanism....33
Search Rules....34
Starting up PRIMOS....35
TCB Protected Resources....37
Protected Subjects....37
Process Creation....39
Protected Objects....40
Files....40
Directories....41
Devices....43
DAC On Tapes....43
Named Semaphores....44
Numbered Semaphores....45
Master File Directories....45
Inter-Process Communication (IPC)....46
Read/Write Concurrency Locks....46
TCB Protection Mechanisms....46
Identification and Authentication....46
Audit....49
Discretionary Access Control....50
Specific ACLs....52
Named ACLs....53
Determining Access....53
Default Access....53
Access Revocation....54
Object Reuse....54
Process Registers....54
Memory Space....54
Physical Disk Records....55

EVALUATION AS A C2 SYSTEM....57
Discretionary Access Control....57
Additional Requirement (B3)....58
Object Reuse....58
Identification and Authentication....59
Audit....60

System Architecture....61
System Integrity....62
Security Testing....63
Security Features User's Guide....64
Trusted Facility Manual....64
Test Documentation....65
Design Documentation....67

Section 4 FUNCTIONAL TESTING....69
PHASE I: System Level Testing....69
Non-accessible Gates....69
LOGIN SERVER System Call Level Test....70
Category 2 Gate Code Review....70
Ring 0 and Invalid Segment Pointers....70
Random Gate Level Tests....71
Object Reuse....72
PHASE II: CPL Level Testing....73
DAC Tests....73
LOGIN SERVER CPL Level Test....73
Testing of the Audit Mechanism....75
PRIMOS Indirection Loop Test....75

Section 5 EVALUATORS' COMMENTS....77
Informal Security Policy Model....77
TCB Interface Control....77
Robustness of the Audit Mechanism....77
Awkwardness of Various Aspects of the DAC Mechanisms....77

EVALUATED HARDWARE COMPONENTS....A-1

EVALUATED SOFTWARE COMPONENTS....B-1

REFERENCES....C-1

ACRONYMS....D-1

EXECUTIVE SUMMARY

The security protection provided by the Prime Computer, Inc. PRIMOS Revision 21.0.1.DODC2 (PRIMOS) operating system, configured as a stand-alone system according to the most secure manner described in the Trusted Facility Manual [5], running on the Prime 50 Series processor, as described on page A-1, "Evaluated Hardware Components", has been examined by the National Computer Security Center (NCSC). The security features of PRIMOS were examined against the requirements specified by the DoD Trusted Computer System Evaluation Criteria (the Criteria) dated December 1985 in order to establish a candidate rating.

The NCSC evaluation team has determined that the highest class at which PRIMOS satisfies all the specified requirements of the Criteria is class C2.

A system that has been rated as being a class C2 system provides a Trusted Computing Base (TCB) that enforces discretionary access control protection and, through the inclusion of audit capabilities, accountability of subjects for the actions they initiate.

PRIMOS is a general purpose operating system running on Prime 50 series processor. These systems include PRIMOS-specific hardware to support the PRIMOS system architecture and protection mechanisms. A large PRIMOS system may be configured to support several hundred users. PRIMOS can be used in a wide variety of environments.

The PRIMOS operating system is a time-sharing system with strong security features. PRIMOS has two basic security mechanisms. The hardware supported protection rings and segmentation provide tightly controlled separate domains of execution. The Access Control Lists (ACLs) provide discretionary access control.

INTRODUCTION

In September 1987, the National Computer Security Center (NCSC) began a formal product evaluation of PRIMOS Revision 21.0.1.DODC2A (PRIMOS), a Prime Computer, Inc. product. The objective of this evaluation was to rate PRIMOS against the DoD Trusted Computer System Evaluation Criteria , and to place it on the Evaluated Products List (EPL) with a rating. This report documents the results of the formal evaluation. This evaluation applies to PRIMOS Revision 21.0.1.DODC2A (which includes the normally separately priced audit utilities) available from Prime in June 1988.

Material for this report was gathered by the NCSC Prime evaluation team, through documentation and interaction with system developers.

Evaluation Process Overview

The Department of Defense Computer Security Center was established in January 1981 to encourage the widespread availability of trusted computer systems for use by facilities processing classified or other sensitive information. In August 1985 the name of the organization was changed to the National Computer Security Center. In order to assist in assessing the degree of trust one could place in a given computer system, the DoD Trusted Computer System Evaluation Criteria was written. The Criteria establishes specific requirements that a computer system must meet in order to achieve a predefined level of trustworthiness. The Criteria levels are arranged hierarchically into four major divisions of protection, each with certain security-relevant characteristics. These divisions are in turn subdivided into classes. To determine the division and class at which all requirements are met by a system, the system must be evaluated against the Criteria by an NCSC evaluation team.

The NCSC performs evaluations of computer products in varying stages of development from initial design to those that are commercially available. Product evaluations consist of a developmental phase and a formal phase. All evaluations begin with the developmental phase. The primary thrust of the developmental phase is an in-depth examination of a manufacturer's design for either a new trusted product or for security enhancements to an existing product. Since the developmental phase is based on design documentation and information supplied by the industry source, it involves no "hands on" use of the system. The developmental phase results in the production of an Initial Product Assessment Report (IPAR). The IPAR documents the evaluation team's understanding of the system based on the information presented by the vendor. Because the IPAR contains proprietary information, distribution is restricted to the vendor and the NCSC.

Products entering the formal phase must be complete security systems. In addition, the release being evaluated must not undergo any additional development. The formal phase is an analysis of the hardware and software components of a system, all system documentation, and a mapping of the security features and assurances to the Criteria. The analysis performed during the formal phase requires "hands on" testing (i.e., functional testing and, if applicable, penetration testing). The formal phase results in the production of a final report and an Evaluated Products List entry. The final report is a summary of the evaluation and includes the EPL rating which indicates the final class at which the product successfully met all Criteria requirements in terms of both features and assurances. The final report and EPL entry are made public.

Document Organization

This report consists of five major sections and four appendices. Section 1 is an introduction. Section 2 provides an overview of the system hardware and software architecture. Section 3 provides a mapping between the requirements specified in the Criteria and the PRIMOS features that fulfill those requirements. Section 4 describes the functional testing approach. Section 5 presents the evaluation team's personal opinions about PRIMOS. Appendices A and B identify specific hardware and software components to which the evaluation applies. Appendix C identifies the references used in this report. Appendix D lists the acronyms used in this report.

SYSTEM OVERVIEW

Background and History

PRIMOS grew from a research project that was funded to study man-machine interfaces. This project resulted in an operating system called TSDOS which ran on the Honeywell DDP 516 machine. This machine supported 16 KBytes of physical memory, a 5 KByte disk drive, and 4 timesharing terminals.

Prime Computer, Inc. was formed to build a machine to run the TSDOS operating system. The idea was to add virtual memory to a low cost timesharing system that was compatible with the 516 and sell it into the 516 user base. The resultant system (P300) became known as the R mode architecture.

Revision 0 of PRIMOS was actually called DOSVM, which supported 32 KBytes of physical memory, up to four 6 MByte cartridge disks, the first Asynchronous Multiplexor Line Controller (AMLC), and a hierarchal file system. It was released in August of 1972.

Soon after the release of the P300, it became apparent that the P300 had limitations that could not be worked around within the then current architecture. The following were what was considered to be the P300s limitations: limited address space (128 KBytes), non-reentrant supervisor living in its own address space, no sharing of user code or data, expensive interrupt/fault handling.

To solve the limitations of the P300, Prime developed the P400. This machine was the first machine to support the V mode architecture that is still the basis of the Prime 50 series systems today. Specifically, the V mode architecture was designed to provide: a large two dimensional address space, an embedded supervisor accessed by a direct call, a natural, pure, re-entrant and recursive environment, sharing of code and data and an efficient context switching mechanism for a multi-user environment.

In August of 1976, Revision 10 of PRIMOS IV (as it was then named) was released as the first support for the V mode P400. In actuality, this version of PRIMOS was like Revision 10 of PRIMOS III (the P300 operating system) with the added capability to run a 2 MByte FORTRAN program. No other V mode features were utilized. The operating system and all users ran as the same process. PRIMOS remained in its own address space, entered by the old SVC mechanism and was still R mode and non-reentrant.

The next stage in the development of PRIMOS saw the gradual increased utilization of the V mode architectural features while preserving compatibility with earlier revisions.

Later revisions of PRIMOS incorporated many sophisticated operating system features demanded by the marketplace. Such features included:

- fully indexed Direct Access Method (DAM) files
- 32 character filenames
- full use of process exchange features
- a reentrant PRIMOS accessible by direct call (PCL)
- 127 file units per user
- the introduction of CPL
- new file system with disk quotas, ACLs, bad block handling, introductions of EPFs
- improved I/O performance - communications facilities
- support for UNIX and SNA, ease of use
- system security features

The current revision of PRIMOS incorporates the above listed features, but some of these features (e.g., UNIX support, networking support) are not included in the evaluated release. PRIMOS is implemented in a variety of languages such as PL/P (Prime's version of PL/1), System Programming Language (SPL), FORTRAN, MODULO II, and assembly (PMA). PRIMOS is 70% PL/P, 15% FORTRAN, 10% PMA, 4% MODULO II and 1% SPL.

Hardware Architecture

The PRIMOS operating system is supported by Prime's 50 series architecture. The 50 series systems are 32 bit virtual memory superminicomputers which multiplex up to 64 Mbytes of physical memory as 512 Mbytes of virtual memory per user. The 50 series systems provide a ring based hardware protection mechanism (see the section beginning on page 20, "Hardware Protection Rings"). All of the CPUs and supporting hardware/firmware are upward and downward compatible systems sharing a common architecture. System security is not affected by implementation differences of this common architecture; these differences only affect the hardware performance of the 50 series processors by implementing such things as a larger cache memory, more user register sets, environmental sensors, instruction pipelining, and emitter coupled logic (ECL) design. In the following discussions of the 50 series architecture (and its varied implementations) it will be shown that the implementation differences do not affect PRIMOS and the system security it offers. Such changes in the 50 series architecture have allowed Prime to provide more computing power, by integrating improved technology into their common architecture.

The following sections describe the hardware components that make up the PRIMOS system, as well as the types of protection mechanisms that are supported in the 50 series systems. The hardware will be covered in enough detail to allow the reader to understand its basic functionality as well as allow the reader to understand the hardware mechanisms used by PRIMOS. The 50 series architecture can be broken down into the following functional areas: the CPU, memory management, process management, and I/O management. These functional areas will be covered in detail; this coverage will also include discussions of the implementation differences of the given mechanisms. Following this discussion, the hardware protection mechanisms and the diagnostics (which show the system is working properly) will be covered.

Major Hardware Components

PRIMOS is supported by four major hardware components whose correct operation is essential to enforce PRIMOS's system security. The four components, CPU, Memory Management Unit, Process Management Unit, and I/O Units will be described in the following subsections. The implementation of the CPU differs throughout the processor family, but these differences are only related to performance. In other words, PRIMOS's TCB is not affected by the differences in hardware components because these differences are invisible at the software level (i.e., the TCB cannot distinguish between the differences implemented in the various high-end hardware components and the implementations of the low-end hardware components). The differences in the CPU's will be shown in the next section.

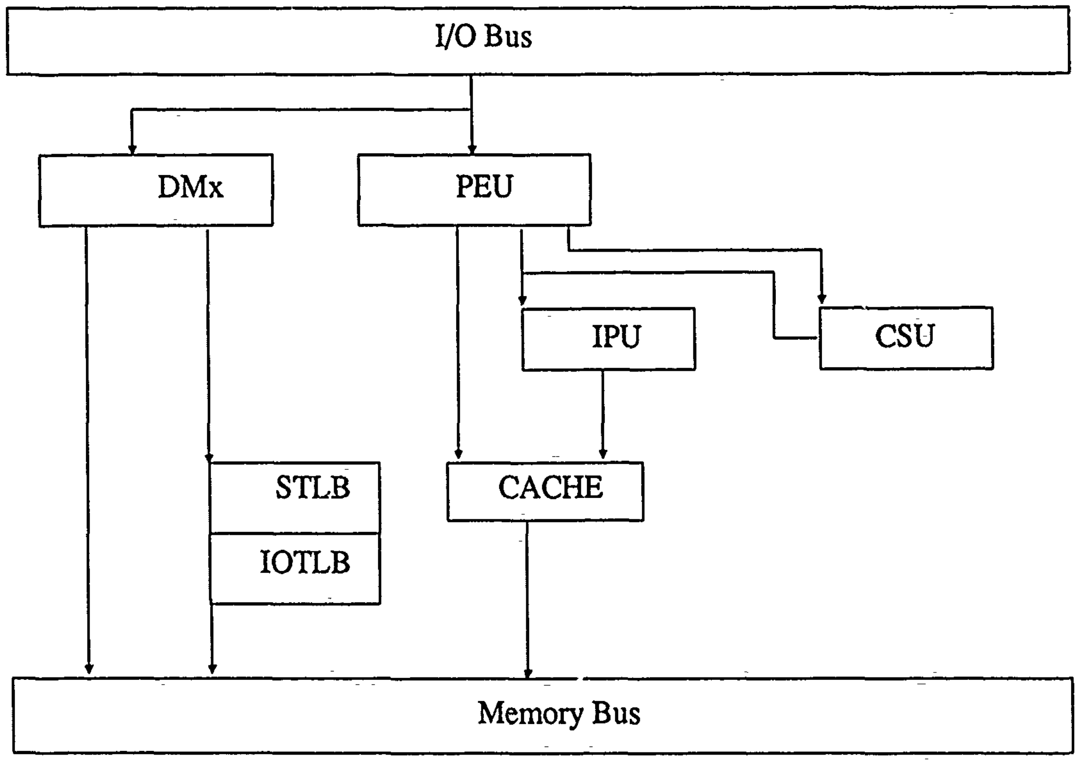
CPU

The 50 series processor family is comprised of the following processors: 2250, 2350, 2450, 2550, 2655, 2755, 9650, 9655, 9750, 9755, 9950, 9955, 9955-II, 6350, 6550. This CPU family is designed around a common architecture with various implementation differences based on the processor model. PRIMOS and its system software performs functionally the same on any of these processors. With the exception of the 6550, all of these CPUs are based on a single-stream architecture. The 6550 processor, is based on a dual-stream architecture and will be described separately.

For the most part, each CPU in the 50 series family can be logically divided into four major units listed as follows:

- cache memory and Segment Table Lookaside Buffer (STLB)
- Input-Output Table Lookaside Buffer (IOTLB)
- Control Store Unit (CSU)
- Processor Execution Unit (PEU).

The 2250, 2350, 2755, 9650, 9655, 9750, 9755, 9950, 9955, and 9955-II systems offer an additional Instruction Preprocessor Unit (IPU) whose primary purpose is to increase the rate of throughput by doing the initial processing on the next two instructions to be executed by the main processor unit. The initial processing is comprised of decoding the next instruction, performing any necessary address calculations, and determining what, if any, registers this instruction will need to access. This preprocessing is performed concurrently with the currently executing instruction, and allows greater execution speed by reducing the work the main processor must perform. The following diagram shows the structure of the basic parts of the single stream CPU.



In the following text, the Cache, STLB, PEU, IPU, and CSU will be discussed. The DMx, and IOTLB will be covered in the I/O management section (see page 17, "I/O Management").

The 50 series systems use the cache storage to speed up the memory reference/address translation process. This is accomplished by using a virtually addressed, write-through cache in tandem with a Segmentation Table Lookaside Buffer (STLB), to speed up virtual to physical address translation. The cache is a data buffer that stores copies of information about the most recently/frequently referenced physical memory locations. A cache entry contains the contents of four bytes of information about recently accessed physical memory. The amount of information found in each cache entry and the number of entries in the cache is based on the type of processor. A generic format of the cache entries for the various processors is given in the following diagram.

V	PPN	DATA

The V bit (validation bit) is used to signify valid data. The Physical Page Number (PPN) specifies the number of the physical page that contains the specified location. The DATA portion is the contents of up to 2 consecutive words of physical memory.

The STLB contains the results of up to 1024 of the last address translations (128 on the 9750 and 9950; 512 on the 2550, 9650, 9955, 9555-II; 1024 on the 6350). The 6350 has a two-set associative STLB that is accessed in parallel to return two STLB entries. An STLB entry is of the following format.

V	M	S	RING 1	RING 3	PID	SEG	PADDR

The V bit indicates the validity of the STLB entry, the M bit determines whether the page has been modified and the S bit indicates sharing (this inhibits the use of the cache). RING 1 and RING 3 are the access right bits. PID is the process ID of the process making the reference. SEG is the virtual segment number and PADDR is the physical page address from an address translation.

When process exchange takes place, all of the entries in the STLB and the cache are invalidated (i.e., the V bit for each entry is turned off) to prevent incorrect information from being used in address translation for the incoming process.

Each of the 50 series CPUs supports a Control Store Unit (CSU). The purpose of the CSU is to speed up execution by implementing many functions in firmware (e.g., procedure calls). Each CPU supports up to 128 KBytes (80 KBytes for the 6350; 50 KBytes for the 9750 to 9955-II; 128 KBytes for the 2755; 64 KBytes for the rest) of on-board firmware address space. For the earlier processors, the CSU is implemented in read-only memory (ROM). Since the microcode on the earlier processors is in ROM, it is not directly modifiable through any of the system interfaces and therefore prevents modification of this microcode. On the later processors, the CSU is implemented in RAM; it is known as the Loadable CSU. The Loadable CSU is loaded with its microcode during a cold start of the system. Once the CSU is loaded, there is no way to modify its contents without bringing the system down and loading a modified version of the microcode. Reloading of the microcode is prevented from occurring through physical security. When the processor is executing this microcode, interrupt processing is handled in such a way as to prevent the interruption of this processing.

This firmware, or microcode, performs tasks on behalf of the running process in response to a trap signal. When such a signal occurs, the CPU begins to execute a predetermined (by the type of trap) section of microcode in the CSU. Control is returned to the running process on completion of execution of the microcode section.

When each microcode instruction is executed, there is a check for external interrupts (see the section beginning on page 17, "I/O Management"). If such an interrupt is present, execution of the next microcode step is postponed while the external interrupt is processed. Control is returned once the interrupt is processed.

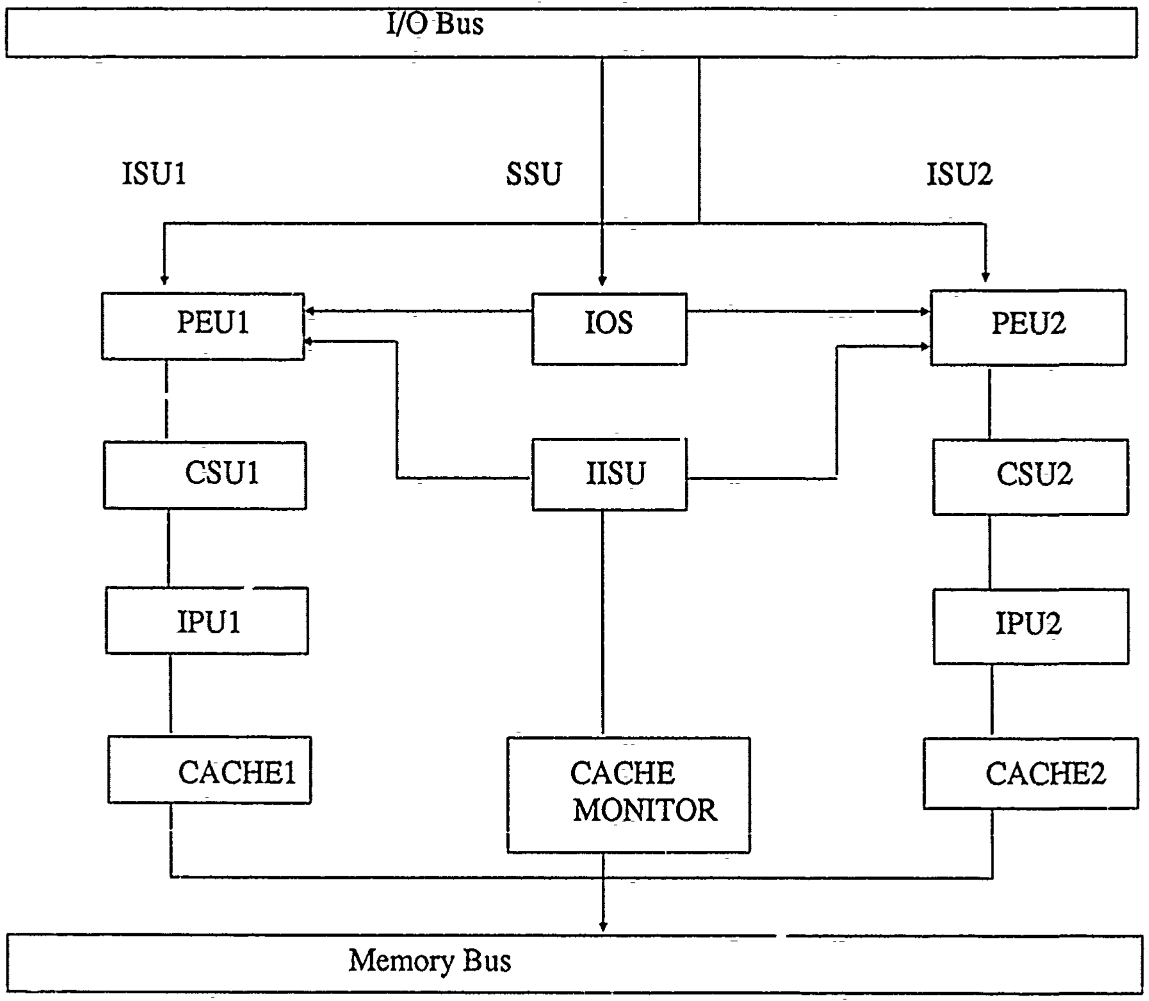
The Processor Execution Unit (PEU) performs the computations required during instruction execution. The PEU is made up of the following components: an Integer Arithmetic Logic Unit (ALU), a Decimal ALU, a Floating Point Unit, a Register File (RF), and a Program Counter (PC). The ALU performs operations on two's complement data. The Decimal ALU and Floating Point Unit perform the operations on decimal and floating point data. The Register File contains up to eleven sets of registers, depending on the processor model. Each set of registers contains 32, 32-bit registers. There are three types of register sets: user, system status and microcode scratch, and DMA. The user register sets contain information about the currently executing process. Specifically, user register sets contain information about the general registers a process can use, addresses of fault handlers, contents of system registers, and other useful information. The 50 series hardware uses another register set to keep track of system status. This register set is invisible to both users and PRIMOS; it is used only by the hardware. There is also a set of registers which provide a read/write area for the 50 series CPU's microcode. The DMA register sets contain registers used by DMA channels to speed up I/O operations.

The RF is configured differently depending on the CPU used. The 9750, 9950, and 9955 CPUs have eight register sets: four sets of user registers, three sets of microcode scratch and system status registers, and one set of DMA registers. The 2550 and 9650 CPUs have twelve register sets: eight sets of user registers, three sets of microcode scratch and system status registers, and one set of DMA registers. The 2250 and earlier CPUs have four register sets: two sets of user registers, one set of microcode scratch and system status registers, and one set of DMA registers.

The benefit of multiple user register sets is best seen by an example. The currently executing process (using one user register set) is interrupted by an incoming process. This incoming process is given a completely different user register set for its execution. This has saved the system from having to copy the first process's register set (state) so the incoming process would have a set of registers to use. Also, when the second process is finished processing its interrupt, the original process does not have to restore its state to resume execution. This mechanism greatly simplifies process exchange by reducing overhead.

For a subset of the 50 series CPUs (i.e., 2250, 2350, 2755, 9650, 9655, 9750, 9755, 9950, 9955, and 9955-II) a special Instruction Preprocessor Unit (IPU) is provided to speed up execution. This unit performs preprocessing of the next two instructions while the main processor execution unit processes the current instruction. Typical work provided by the IPU involves decoding instructions, address calculation, determining what registers will be used by the next instruction, and fetching the second "next" instruction. By use of the IPU, the PEU is able to execute the next instruction without delay. This unit is only providing assistance to the PEU; PRIMOS does not even know of its existence. Since PRIMOS cannot tell whether it is running on a processor with an IPU or a processor with no IPU, the presence of an IPU will not affect the normal operation of PRIMOS.

The dual stream architecture is implemented by the 6550 processor. The following diagram shows the basic dual stream architecture:



This dual stream architecture is realized by two Instruction Stream Units (ISU), which each have the full capabilities of a single CPU. Each ISU has its own PEU, IPU, CSU, cache, STLB, and IOTLB. Both of the ISUs share a common memory and I/O bus. Each of the ISU's executes an independent stream of instructions. This stream is synchronized by a Stream Synchronization Unit (SSU). The IISU manages the inter-ISU communication between ISU1 and ISU2.

The SSU is primarily responsible for preventing improper information from being loaded into the cache of the ISUs. To accomplish this the SSU keeps a list of the contents of each ISU's cache in its cache monitor. When data is to be loaded into either cache, the SSU checks to see if the incoming data will invalidate any entries in the other ISU's cache; if so, the appropriate ISU is informed and it updates its cache accordingly. The SSU is also responsible for coordinating memory accesses for memory and I/O. This is necessary because the ISUs share the same main memory and copy of PRIMOS. The coordination of these resources is accomplished by four locking mechanisms (process exchange lock, queue lock, check lock, mutual exclusion lock); whose correct operation ensure proper execution of the two input streams of instructions. Diagnostic operations and communications between ISUs are also handled by the SSU.

The 50 series CPUs support a verbose instruction set. There are 588 instructions implementing four addressing modes (see the section beginning on page 12, "Memory Management"), and supporting several data representations (i.e., Fixed-point data, Floating-point data, Decimal integers, Character strings and Queues). Of the 588 instructions that make up the 50 series instruction set 84 are restricted (see the section beginning on page 23, "Privileged CPU Instructions").

The 50 series CPUs handle hardware faults, machine checks and device interrupts. A fault is an unexpected event which occurs in the scope of the current executing process and is detected by the hardware. Once a fault is detected, its type is determined and the appropriate software fault handler is called. Valid fault types are:

- Restricted Instruction
- Process
- Page
- SVC
- Unimplemented Instruction
- Semaphore Overflow

- Illegal Instruction
- Access Violation
- Arithmetic Exception
- Stack Overflow
- Segment
- Pointer

Machine checks are generated by environmental conditions such as the over-heating of the CPU cabinet. Machine checks are discussed under the topic of hardware diagnostics (see the section beginning on page 23, "Hardware Diagnostics").

There are two primary types of interrupts: hardware and software. Hardware interrupts are generated by devices for data and control information transfer (see the section beginning on page 17, "I/O Management"). Software interrupts are generated by the INEC instruction. The INEC instruction is used by the phantom interrupt code to restore the interrupted process after a hardware interrupt. It is responsible for reloading the PB and KEYS, issuing a CAI instruction to clean up the I/O bus, enable interrupts, and to notify the appropriate semaphore (or wake up the interrupt server, IPQIPC).[1]

Memory Management

Since PRIMOS is a virtual memory system, proper memory management is of utmost importance. As with any virtual memory system, PRIMOS provides a moderately large (512 Mbytes) per user virtual address space from a smaller (8-64 Mbytes) physical address space. The mapping of physical to virtual memory and the coordination of this process is, in part, handled by the 50 series hardware. The following section will describe what parts of the hardware are involved in memory management and what tasks they perform.

The key area of memory management handled by the 50 series hardware is the translation of virtual segment addresses to physical page addresses. This work is performed by firmware in the 50 series system's CSU. Before the address translation process can be explained, the data structures used in

1 IPQIPC is a code path that coordinates interrupt processing.

this translation will be described. These data structures are the STLB, Descriptor Table Address Registers (DTARs), Segment Descriptor Tables (SDTs), and either Page Map Tables (PMTs) on the 9750, 9755, 9950, and 9955 processors, or Hardware Page Map Tables (HMAPs) on earlier processors. Since the STLB has already been covered it will not be discussed in any further detail.

There are four DTARs (DTAR0 through DTAR3) used to reference the four groups of a user's virtual address space (see page 19, "Virtual Memory Segments"); shared segment groups are referenced through DTAR0 and DTAR1, and the unshared segment groups are referenced through DTAR2 and DTAR3. A DTAR is of the following format:

SIZE	A	-	B

bit 1 10 11 16 17 18 32

SIZE specifies 1024 minus the size of the SDT. The concatenation of A and B specify the SDTs physical memory address. Each of these four DTARs points to a SDT.

An SDT contains from 1 to 1024 32-bit entries called Segment Descriptor Words (SDWs). Each SDW describes one segment . . the given region. An SDW is of the following format:

ADDRESS	F	A1	-	A3	ADDRESS

bit 1 16 17 18 20 24 26 27 32

A segment (see page 19, "Virtual Memory Segments"), is described in the SDW by the location of its page map table (derived from the two ADDRESS fields), a fault bit (F), and the access rights associated with the given segment (A1 and A3).

The PMT and HMAP describe physical pages of memory associated with a given segment. The primary difference between the PMT and HMAP is that the PMT has a 32 bit entry length and the HMAP has a 16 bit entry length. The structure of the PMT and HMT are as follows:

R	U	M	S		PAGE ADDRESS	PMT

bit 1 2 3 4 5 16 17 32

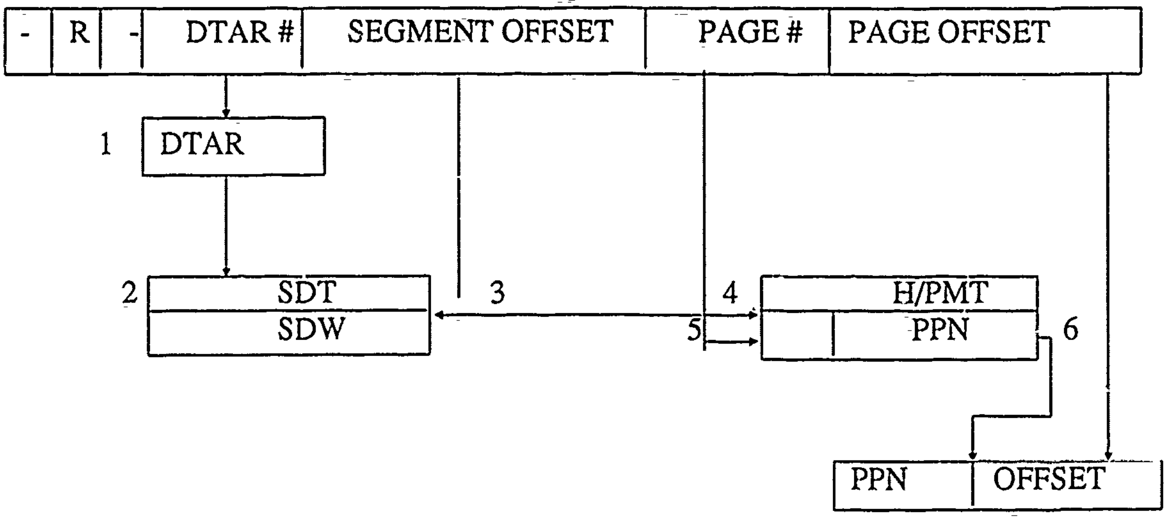
R	U	M	S	PAGE ADDESS	HMT

bit 1 2 3 4 5 16

The R bit is the resident bit, the U bit is the used bit, the M bit is the modified bit, and the S bit is the shared bit. The PAGE ADDRESS is the high order bits of the physical page address. Older processors only supported 8 MB of physical memory, while new ones can support up to 64 MB of physical memory.

The heart of address translation is the virtual address. A virtual address is a 32 bit word which provides the associated physical page's location. Its format is given in the address translation diagram. To translate a virtual address to a physical memory location the CPU must perform at most three steps: Check the STLB for an already computed address translation (if found access the cache), possibly translate the virtual address into a physical memory address, and possibly find the correct virtual page on disk and move it into main memory. The first task is done by hardware, the second by firmware, and the third by a software page fault handler. In the following text the first two steps will be covered; the third step is covered in the software architecture section (see page 25, "The PRIMOS Kernel").

Address translation is initiated by a process's request for a virtual address. Once a virtual address has been requested, the currently executing process is trapped into the address translation microcode. This microcode, known as the Address Translation Mechanism (ATM), uses the previously described data structures in the interpretation of the virtual address as depicted in the diagram below.



The ATM first checks the STLB to see if the address has already been calculated. If the address is in the STLB, then this value is returned as the requested address translation. If the appropriate entry is not found in the STLB, address translation is performed. The steps performed in address translation are as follows:

1. Select the DTAR specified in the DTAR number field.

2. Use the address in this DTAR to reference the appropriate SDT.

3. Use the segment offset to reference the appropriate SWD in this SDT.

4. Use the address in this SWD to reference the appropriate mapping table (PMT or HMT).

5. Use the page number to reference the appropriate mapping table entry.

6. If the page referenced by this entry is in physical memory, form the Physical Page Number by concatenating the PPN from the mapping entry with the page offset from the virtual address.

These steps result in a physical address derived from the virtual address provided to the ATM. With a valid physical address the STLB is then updated. Once the STLB contains a valid address, the hardware determines the access rights that will govern the reference (see the section beginning on page 20, "Hardware Protection Rings"). If the access check is successful, the hardware will access the cache and perform the memory reference. If it is unsuccessful, a software break (i.e., check) occurs. This check is handled by the microcoded interrupt handler in the CSU outside the context of the currently executing process. When this processing is complete, an error is returned to the process which made the access violation.

The 50 series CPUs support four addressing modes, each of which forms addresses differently. The three useful modes are V-mode (virtual addressing), I-mode (register), and R-mode (relative addressing). The fourth mode, S-mode (sectored addressing) is still supported for compatibility reasons. V-mode performs short and long addressing operations. It has a wide variety of registers to use. I-mode is similar to V-mode except that registers are used more heavily to provide faster accesses. R and S mode addressing are both sector (512 words) based. Addresses in these modes are related to the memory sector that is being accessed. S-mode is a hold over from when Prime used Honeywell 316 and 516 hardware. When executing in S-mode the 50 series processor emulates these type of processors.

Process Management

Process management on the 50 series machines is realized by the Process Exchange Mechanism (PXM). The PXM is responsible for determining which process to run next, saving the state of the outgoing process, clearing the cache and STLB, locating a register set for the incoming process, restoring the state of the incoming process, and transferring of control to the incoming process. The mechanisms used in performing these tasks are implemented in the 50 series's microcode (CSU).

The PXM is made up of six primary elements; three data structures, two PXM instructions, and the dispatcher. The three data structures are the Process Control Blocks (PCBs), the Ready List, and the Wait list. The two PXM instructions are WAIT and NOTIFY. In addition to these primary elements, PXM also manipulates the RF and process interval timer during process exchange.

A PCB specifies the state of a given process where the state is made up of the contents of:

- its register sets,
- timers,
- pointers to virtual memory,
- fault information,
- control and processor information,
- the priority level, and, - for the 6550 processor, which of the two ISUs should run next.

Also contained in the PCB are the process abort flags. These flags are used to signal the occurrence of asynchronous events. All of the PCBs for the running processes on the system are kept in a single dedicated segment. Because of this fact, there is a physical limit of 256 PCBs.

The Ready List is a list of processes ready to execute. This list is ordered based on the priority of the processes in the list. The elements of the ready list are a series of headers that make up the ready list, a data base of PCBs, and two registers (PPA and PPB) used to point to PCBs. The ready list itself is made up of headers, one for each level of priority. These headers are allocated in contiguous memory locations. Each header is associated with a PCB. The ready list data base is made up of a linked list of PCBs whose associated processes are ready to execute. There is a separate list for each priority level. PPA points to the currently executing process's PCB and PPB points to the PCB of the process that will execute next.

The Wait List is a list of lists of processes waiting for an event to occur. Each wait list is associated with a linked list of PCBs and a semaphore for each linked list. The semaphore is used to wake up the process that is waiting at the front of each list. Semaphores are hardware implementations of Edsger Dijkstra's "P" and "V" operations [1].

Moving processes from the Wait list to the Ready list is accomplished with the WAIT and NOTIFY instructions. These instructions are restricted and therefore they can only be executed from ring 0. The WAIT instruction checks the status of the event the process is waiting for. If the event has occurred the process is allowed to stay running. If it has not occurred WAIT blocks the running process and moves it to the appropriate Wait list. NOTIFY is implemented as two instructions (NFYE and NFYB) whose difference is only in the queuing algorithm used. They both are used to remove the process at the head of one of the Wait lists, when a given event occurs. The process is then moved to the Ready list.

The Dispatcher is the mechanism that transfers control of the CPU between processes. It is a firmware implemented routine responsible for deciding which process to run next, assigning a register set to that process, managing the RF, and turning the process timer on and off. The actual transfer of control occurs when the Process Interval Timer (PIT) signals the Dispatcher that the current process's time is up. The Dispatcher will first turn off the PIT and then save the current process's state onto the Ready List. It then forces the process at the head of the Ready List (i.e., the process with the highest priority) to begin execution. The Dispatcher is also responsible for allocating the incoming process's user register set.

I/O Management

I/O management on the 50 series CPUs is divided into two types: programmed I/O (PIO), and direct memory I/O (DMx). This division is based on the actions performed to initiate and carry out the given type of I/O. PIO occurs when a process requests the processor to issue a request to a particular controller to perform a given I/O operation (e.g., send control information to a device, move data between the device and the CPU). All of the work in PIO transfers is directed by the CPU (i.e., it's CPU instruction based). DMx transfers are also initiated by the CPU, but the given controller performs the real work of the transfer (i.e., after being asked by the CPU to perform the transfer, the controller directly accesses main memory rather than using CPU instructions).

PIO is CPU instruction based I/O. All of the instructions used in PIO (INA, OCP, OTA, SKS) are restricted instructions. These instructions are used to send small packets of data between the CPU and devices. Since these instructions are restricted, they can only be executed by ring 0 procedures. This limitation prevents normal PRIMOS users from directly accessing devices and therefore only allows PIO operations for trusted ring 0 processes.

DMx is used for multiple data transfers. To accomplish this, DMx operations allow devices to access memory directly. This access is controlled by both the hardware/firmware of the device controller (i.e., it is only allowed to access a predetermined, bounded area of memory), and by the CPU (i.e., it initiates the transfer and loads the information that specifies the bounded region of main memory). DMx operations are prevented from circumventing system security by using a defined and trusted means of transfer (i.e., the firmware is assured to perform properly, the CPU is assured to perform properly, and the algorithm used in DMx transfers is well defined and is governed by the hardware protection mechanisms) of data and by only allowing the given controller to perform actions the CPU has allowed.

DMx transfers occur without software intervention. In performing a DMx transfer, a temporary diversion in the sequence of microcode from CPU to DMx transfer routines occur. This transfer of control is known as cycle stealing. At the end of the transfer the CPU microcode is allowed to continue where it was interrupted. Since the transfer of control takes place after the execution of the CPU microcode instruction when the trap was sensed, and since the DMx mechanism uses its own transfer buffers (IOTLB) and register set, the 'currently' executing process is completely unaffected by the operation.

There are four types of DMx transfers: DMA, DMC, DMT, and DMQ. Each method differs in terms of speed, volume of data, and control features. The following table summarizes the steps involved in a DMx transfer.

1. Controller raises a DMx request.

2. Since the CPU scans the backplane for DMx requests at the end of each microcode instruction, it will see and process the DMx request.

3. If this request is the highest hardware priority (based on backplane location), it will be processed.

4. Controller sends the CPU its channel address (acquired during cold start) and indicates the transfer direction and type.

5. The DMx microcode performs the transfer, and checks for more DMx requests. If there are any, they are processed.

6. This process continues until there are no more requests, at which time control is restored to the interrupted process.

DMA is primarily used by disk, tape and PNC controllers, DMC is used by MDLC, SMLC, AMLC, QAMLC, and other controllers, DMQ is used by QAMLC controllers and to down-line load ICS microcode (described later in this section), and DMT transfers are used by the SMD and ICOP controllers. DMx operations do not affect system security if the firmware that drives the controllers works as specified. With this assumption, it must be shown that the firmware is handled in such a manner that it cannot be tampered with by PRIMOS users. All of PRIMOS's controller specific firmware is stored in ACL protected directories of the PRIMOS file system. Since these directories are protected by the ACL mechanisms, as directed in the TFM [5], the system prevents unauthorized access to these files (i.e., they are tamper-proof).

The process of loading the firmware into the given controller (down-line loading or hereafter DLL) is handled by a ring 0 kernel process during a cold-start of PRIMOS. The DLL process is responsible for determining what firmware goes to which controller and for performing the DLL of the appropriate firmware to a given controller. Since the mechanisms that perform this task are part of the TCB, their correct operation is assured.

Hardware Protection Mechanisms

This section describes PRIMOS's hardware protection mechanisms which are visible to the system's users. Those protection mechanisms not user-visible (i.e., those mechanisms which detect and handle system malfunctions) will be covered in the section beginning on page 23, "Hardware Diagnostics".

Virtual Memory Segments

Virtual memory is divided into 128 KByte units known as segments. These virtual segments allow users and PRIMOS to organize their virtual address space by providing each process with 4096 possible active segments. These possible active segments are in one of four groups of 1024 segments each. The first 2 groups comprise a shared region which typically contains PRIMOS, shared libraries, and shared subsystems. Since each user shares a copy of PRIMOS, the O/S interface can be handled much more cleanly than in earlier versions of PRIMOS (i.e., no interrupts, supervisor calls, or system traps are necessary when a user requests a service of PRIMOS). The code found in this region is unmodifiable (or pure code) and therefore does not allow users to transfer information through this region. The other two groups of segments provide each process with a completely private virtual address space. All of virtual memory is governed by the memory management facility (see the section beginning on page 12, "Memory Management"); ensuring that users cannot access/modify data not found in their virtual address space.

Hardware Protection Rings

The Series 50 architecture provides three hardware implemented protection rings (0, 1, 3), of which PRIMOS uses two (0 and 3). The ring mechanism provides a simple, effective way to protect the vital parts of the system on a per-segment basis. Each ring represents one of three levels of protection for PRIMOS and its users. Ring 0 is the highest level of protection and therefore contains all of the security critical mechanisms of PRIMOS (i.e., its kernel). PRIMOS's TCB runs under ring 0 protection, which means that its segments cannot be accessed by a user except through protected entrypoints, and that PRIMOS has read, write, and execute privileges to all segments. Ring 0 therefore has unrestricted access to all memory. Ring 3 is the lowest level of protection and therefore is limited to a specific subset of memory (defined by PRIMOS). Each segment under ring 3 protection may have different combinations of read, write, and execute access rights. User processes or applications always run in ring 3. Ring 1 is currently not used by PRIMOS; it is reserved for possible future expansions of PRIMOS. The protection mechanism provided by rings ensures that a user can only access the kernel through protected entry points known as "gates".

Access control on the segment level is determined by the ring access rights found in the Segment Descriptor Word (SDW) for each segment. This information is used during address translation, to determine if the running process has access to a given segment. When the currently running process attempts to access an address in memory, the STLB is checked to see if it contains information on the given address translation. If it does not, a microcoded part of PRIMOS (contained in the CSU) called the Address Translation Mechanism (ATM) must perform the address translation. The first step performed by the ATM in address translation, is to generate the "effective" ring number. A section of hardware forms the ring number for the address translation firmware. This is accomplished by logically OR'ing the current ring number, taken from the Program Counter (PC), and the ring number contained in the virtual address. This process is called "weakening" the ring number. By weakening the ring number it is ensured that the highest ring number of the PC/virtual address pair is used in address translation. This in turn, ensures that inner ring software always uses the access rights of the process that formed the address.

Once the effective ring number is determined, the addressing firmware references the current SDW in the Segment Descriptor Table (SDT). This process is accomplished by referencing the DTAR field in the virtual address, which designates a register that points to a SDT. The offset in the SDT to the desired SDW is also provided in the virtual address. If the desired SDW is found in the SDT, then the addressing firmware checks the current process's access to the segment in question. This access control decision is based on the contents of the three bit field in the SDW for the given ring number (000 = no access, 001 = gate access, 010 = read access, 011 = read and write access, 100 and 101 reserved, 110 = read and execute access, 111 = read write and execute access) and the effective ring number provided by the ring weakening process. Hardware selects the appropriate access rights to the given segment based on the effective ring number and the ring number for the given segment (found in the SDW for that segment). It is this hardware that detects access violations. If the access is determined to be invalid, then an access violation occurs; otherwise address translation continues. A more detailed description of address translation can be found in the "System Architecture Reference Guide" [6].

The previous sequence of events occurs at every segment access. Through this mechanism, PRIMOS ensures that outer ring processes cannot arbitrarily access inner ring (or the same ring) segments without the process possessing proper access rights. This same mechanism also provides a controlled means of accessing inner ring segments from outer rings. This process is known as ring changing and will be described next.

In PRIMOS there are some inner ring procedures that outer ring procedures might want to call. Since the normal read, write, and execute access rights will not allow such inward references, these inner ring procedures must specify a special access right called "gate access". Gate access allows ring 3 procedures to access PRIMOS's inner ring procedures without harming the rest of the system.

To change the ring of execution in PRIMOS from a higher ring to a lower ring (known as a direct entrance call), an access control decision must be made to determine if the currently running process has access to execute the lower ring procedure (i.e., the lower level procedure must have its gate access bits set in its SDW). Since inward calls pose a possible security problem, PRIMOS has all of its user accessible ring 0 procedures set apart from the rest of the system software. This is accomplished by having all of the Entry Control Blocks (ECBs) of these procedures located in a special gate access segment. A process can only call those lower ring procedures found in the gate access segment. The actual change of rings of execution is accomplished by changing the ring of execution bits contained in the Program Counter (PC). This occurs whenever a PCL or a PRTN instruction is executed on a gate.

When a process calls a procedure whose ECB is in the gate access segment, it must be a valid gate access procedure. All valid gate access procedures are defined in a ring 0 table known as the gate table. All of the valid gate access procedures define the set of trusted entry points into ring 0. If the process calls a valid gate access procedure, PRIMOS allows access to the ring 0 procedure, and the effective ring of execution is changed (via PCL). Control is returned to the caller on completion of the called procedure (via PRTN). By use of this mechanism, PRIMOS has effectively protected those portions of the TCB that are not user accessible from being accessed by means of the gate access segment.

When the PCL instruction is executed the following sequence of events occur:

1. Calculate the callee's ring number
2. Allocate a new stack frame for the callee
3. Save the caller's state
4. Load the callee's state
5. Calculate and store indirect pointers for the callee's use

When PCL begins execution, it calculates the ring number of the call. PCL looks at the appropriate STLB entry, since it contains access rights for the calling procedure. PCL uses these access rights to determine if the caller has access to the callee's ECB. If the STLB specifies read access, PCL weakens the ring number contained in the callee's ring field to that of the caller. If the callee's ECB is in the gate segment, PCL uses the ring field contained in the callee's ECB as the ring number. Once the ring number has been calculated, PCL allocates a new stack frame for the caller, saves the caller's state, and loads the callee's state into the newly allocated stack frame.

If the callee expects arguments, its first instruction must be an ARGT instruction. If the callee is expecting arguments and the ARGT instruction is not the first instruction no argument passing will take place; no argument will be passed if ARGT is in any other location other than the first. If ARGT is the first instruction, several pointers to the arguments would follow the PCL instruction. These pointers point to argument templates which contain directions that PCL uses to form indirect pointers to the actual arguments. Indirect pointers are saved in the stack frame that the callee uses to reference the arguments.

Several argument templates may be used in succession to form one indirect pointer. One template may specify a level of indirection; the next, a base register. Each template contains an S bit that determines if that template is the last one to be used to form a single indirect pointer. If the S bit contains a 1, then the argument is the last one to be used for the given indirect pointer, and the processor will store it into the current stack frame. An S bit of 0 means there are more templates to be processed.

Each template also contains an L bit to indicate if it is the last one for the last indirect pointer. When L and S are both 1, then the given argument is the last one for the last pointer. When L is 0, other arguments follow it. When L is 1 and S is 0, the processor forms dummy indirect pointers.

Indirect pointers are formed by PCL after the caller's state is set up. To form an indirect pointer, PCL first forms the effective ring number by "weakening" the ring numbers in the PC and base register. The segment field of the caller's base register is stored in the indirect pointer's segment field and the effective ring number is stored in the ring field. Offsets are computed and stored in the offset field of the indirect pointer. The indirect pointers are then stored in the caller's stack frame.

If S contains a 1, PCL stores the calculated indirect pointer in the next stack frame location. If L also contains a 1, then there are no more indirect pointers to be calculated. A 0 in L indicates that there are more arguments to follow, so PCL proceeds with the next one.

If the number of indirect pointers produced is greater than the number the callee expects, PCL ignores the extras. If the number of pointers produced is less than what is expected, PCL creates dummy pointers and stores them in the current frame. The callee can reference the dummy pointers only to pass them on to other new procedures; if such a reference occurs, the new procedure will see these pointers as being omitted. Any use of omitted pointers other than to pass it on causes a pointer fault.

The last thing PCL does is to transfer control to the called procedure. The last instruction in this procedure must be a PRTN. When PRTN is executed, control is transferred back to the caller and execution begins with the instruction following the PCL instruction. PRTN also is responsible for deallocating the stack frame created when the procedure call was first made, and returns the caller's environment. The latter is accomplished by loading the PC with the appropriate address and ring number which is determined by weakening the saved PC's ring number and the current ring number. This prevents inward returns, yet allows returns from gated calls to work properly.

Privileged CPU Instructions

Certain CPU instructions are privileged in the sense that they can only be executed in ring 0. This set of reserved instructions is designated by their opcodes, which are detected by the hardware at time of execution. Once an opcode is determined to be a privileged instruction, the current ring of execution bits in the PC are checked. If the currently executing process is executing in ring 0, the instruction is executed. If a privileged instruction is executed in any ring other than ring 0, a restricted mode fault will result. This form of protection is guaranteed by the hardware and therefore cannot be altered at the software level.

Hardware Diagnostics

All Prime 50 series systems have on-board microverification programs as a standard feature. These microprograms execute each time the system is cold-start booted. Their purpose is to test all of the hardware portions of the system to ensure that they are working properly. These tests are not security specific, but since PRIMOS's system security depends on the correct operation of the hardware, these tests are important to ensure that PRIMOS will be comming up in a "safe" environment (i.e., the hardware will perform as PRIMOS expects it to). All failures discovered by these programs are reported to the system console for further action. If the problem is critical, then PRIMOS will not continue the boot process until the problems are fixed.

The 2550, 9650, 9655, 9750, 9755, 9950, and 9955 CPUs provide several features that increase the efficiency and reliability of PRIMOS. Common to all of these processors is the Uninteruptable Power Source (UPS) and Environmental Sensors. These features are supported by a diagnostic processor system that supports inputs from the UPS and Environmental Sensors. This subsystem allows the CPU to be brought to an orderly shutdown in the event of an over-temperature or a main AC power loss with messages being output to the supervisor terminal. To conserve power, this diagnostic processor does not accept typed commands during system shutdown or while the UPS is active.

The UPS uses two signals: UPS Active, and UPS Battery Low. UPS Active means that the main AC power has been interrupted. UPS Low Battery means that five to six minutes remain before system power is lost. When the UPS is powering the entire system and a battery low condition occurs, the diagnostic processor sends a processor check to the CPU and waits up to five minutes before powering down the system. When a UPS active condition occurs and the UPS is powering only the CPU, memory, and diagnostic processor, the diagnostic processor sends a power failure signal to the CPU, causing the CPU to log the power fail condition and then halt. When the UPS condition changes from active to inactive, this change implies that the AC power has returned. The diagnostic processor initiates a warm start to the CPU provided that the operator keyed a WARMON command at the supervisor terminal before the main AC power was lost. Otherwise, the diagnostic processor initiates a master clear and enters control panel mode at the system console. At this time logins are disabled.

There are three environmental sensors: a cabinet overtemperature sensor, a processor overtemperature sensor, and an air flow sensor. When any of these sensors reports a problem to the diagnostic processor, it in turn sends the CPU an environmental check. The CPU will handle the check accordingly.

Software Architecture

This section describes PRIMOS's TCB architecture. This architecture is based on the privilege ring mechanism as described in the section beginning on page 20, "Hardware Protection Rings".

The software TCB for the PRIMOS operating system consists of the following components:

1. all of the code in ring 0, accessible through gated procedure calls (see page 32, "Use of the Gate Mechanism") This ring 0 code accessible through gated procedure calls or interrupts constitute the kernel of the PRIMOS operating system.

2. processes executing within ring 0, and

3. certain trusted system processes which execute in ring 3 (see page 28, "Ring 3 Trusted Processes").

These three components of the software TCB are described in greater detail in this section.

The PRIMOS Kernel

The PRIMOS kernel consists of all the code executed by gated procedures or interrupt handling in order to provide implicitly or explicitly requested user services. The major services provided are listed below:

1. demand paging,

2. memory management (page allocation, dynamic memory allocation),

3. process management,

4. scheduling (maintaining scheduling queues),

5. file and directory management (different file structures),

6. ACL handling,

7. device drivers (including asynchronous I/O capabilities),

8. interrupt handling (phantom interrupt code),

9. fault handling (condition mechanism),

10. system call management (parameter passing and checking),

11. the Locate buffer mechanism (LRU disk buffer cache),

12. semaphores, and

13. system locks.

Some of these functions are described in more detail below. The demand paging mechanism under PRIMOS is controlled by the procedure PAGTUR, whose function is to handle all page faults, and correctly provide object reuse at the page frame level. Memory management code allocates both pages, and smaller areas of memory for both users and PRIMOS.

Process management code works in conjunction with the Login Server process to create the necessary data structures required to support a process under PRIMOS (see page 16, "Process Management"). Once a process is created, there is code associated with the scheduler to maintain the scheduler queues other than the ready list (which is maintained by the hardware).

File and directory management code and ACL handling code manage the use of files and control the utilization of disks (see page 57, "Discretionary Access Control"). This code allows for the use of several different file structures ranging from contiguous files to indexed sequential access method files (ISAM).

Device drivers and interrupt handling code allow for the control of peripheral devices, and internal faults (see page 17, "I/O Management"). Fault handling is used to provide the condition mechanism which is an implementation of software interrupts.

One service provided by PRIMOS is the Locate Buffer mechanism which was designed to increase I/O efficiency. The Locate Buffer mechanism is an LRU based disk buffer cache which keeps the most recently used disk blocks memory resident and flushes out the least recently used modified disk blocks. For every disk I/O request, PRIMOS will look for the requested block within the buffer cache, before actually accessing the disk.

PRIMOS also provides named or numbered semaphores and system locks for process synchronization and file record locking. System locks are implemented through the use of semaphores.

Ring 0 System Processes

The following processes execute within ring 0:

Process Name	Description
Timer Server	Provides time keeping services for PRIMOS
Copy Server	Provides disk mirroring capabilities for PRIMOS

The Timer Server

The timer server (TIMER_PROCESS) provides PRIMOS and its users, with all of the timing related system services. This includes time of day, process timeout timing, and individual process defined timers. The timer server itself is a separate high priority process executing in ring 0. It comes into being at cold start time and must be ready to execute at all times. The interface to TIMER_PROCESS is implemented in a set of ring 0 gates. TIMER_PROCESS itself is not security critical; rather it is system critical and therefore defined as part of the TCB. This server makes no access control decisions, but its correct operation is necessary for other system functions to perform as specified.

The Copy Server

The copy server is a process that provides a disk mirroring capability to PRIMOS. Disk mirroring consists of duplicating all writes to a primary disk on a secondary disk. Disk reads are not duplicated; rather the reads are split so that half of the reads are from the primary disk and half from the secondary disk. This reduces the average seek time.

A mirrored disk is created by calling a gate (CUCPY$) that repeatedly reads a record from the primary disk and writes it to the secondary disk. Tne copy server is brought into being by use of the SPAWN$ gate in response to a MIRROR_ON or the appropriate configuration directive.

Disk mirroring can be initiated at system start-up by using the configuration directives, or while the system is running by using the MIRROR_ON command. This service can be turned off at any time with the MIRROR_OFF command. All of these commands are privileged commands and are executable at the system console. The copy server itself runs as a ring 0 process. When writing to a mirrored disk, the copy server itself is responsible for making the copies of a write to both the primary and secondary disks.

Ring 3 Trusted Processes

PRIMOS uses a small set of dedicated processes (servers) to perform certain functions such as login, auditing, error logging, spooling user input and output, and batch service. These servers are security relevant and they are discussed to provide a complete description of the system. These processes need to be trusted because of the functions they perform and because they are in some sense of the word, privileged.

The following processes execute in ring 3:

Process Name	Description
Log in Server	Provides the log in/log out interface for PRIMOS
Audit Server	Provides the auditing capabilities for PRIMOS
Distributed System Management Servers	Provides several functions for PRIMOS
Spooler Server	Provides spooling capabilities for PRIMOS
Batch Server	Provides batch services for PRIMOS

The Login Server

Another highly user visible subsystem (process and associated gated procedures) is the login server (LOGIN_SERVER). This subsystem is started during the initial cold start of PRIMOS, and is responsible for processing and validating login requests. To perform this validation task, LOGIN_SERVER accesses the System Administrator's Directory (SAD), a protected object, and therefore must be part of the TCB. The inclusion of the login server in the TCB is based on its process type (U$LSR). This server also provides the initial identification and authentication (I&A) of PRIMOS users (see page 46, "Identification and Authentication") necessary for a C2 system. LOGIN_SERVER itself executes in ring 3 but is supported by a private set of ring 0 gates. These gates provide the login server and its related commands with a controlled interface into PRIMOS. Access to these gates is limited to LOGIN_SERVER's process type (U$LSR) and to user 1. User 1 is the process which comes up on the system console. A more detailed description of process types is covered on page 37, "TCB Protected Resources".

Closely related to the login server are the user specific password functions. These functions perform the tasks of setting a user password, changing a user password, and other password manipulation functions. The gates used by the command level functions which perform these tasks are all ring 0 gated procedures. They provide a trusted means of performing such actions as changing and encrypting passwords. These gates are not part of the login server's set of gates and therefore can be accessed more freely than the login server gates.

The Audit Server

The audit server (AUDITOR) is a separate ring 0 process responsible for collecting audit data and moving this data from memory buffers out onto external storage media. AUDITOR is started up when the security auditing facility is first enabled by the SECURITY_MONITOR -START command and is disabled by the SECURITY_MONITOR -STOP command (see the section beginning on page 49, "Audit").

When AUDITOR is initially started up, a number of memory buffers are allocated to temporarily store audit data until it is written to external storage media. Audit buffers are 4K bytes (2 physical pages) in length; a minimum of 2 and a maximum of 12 (default 4) audit buffers may be configured while the audit mechanisms is running. The number of buffers is set up at initialization time and can be changed while AUDITOR is up and running. If the audit data buffers fill up before AUDITOR flushes them, then the processes writing into these buffers are forced to WAIT until the problem is remedied.

AUDITOR itself does not perform audit data collection. This is accomplished by the use of audit PROBES. A PROBE is essentially a call to AUDITOR requesting it to record the data describing the occurance of the given event. PROBE calls are coded into those procedures that are auditable. If a running process is auditable (i.e., there is a PROBE call encoded into the given procedure) and being audited (i.e., the System Administrator is auditing the given event) a call to PROBE will cause information about the caller of PROBE to be written into the audit data buffers by AUDITOR. Auditable events are covered in the section beginning on page 49, "Audit".

Distributed System Management (DSM) Servers

The DSM facility under PRIMOS is primarily used in a networking environment but is also usable in a PRIMOS installation which is a single node configuration for the purpose of hardware event logging. A C2 certified site uses DSM solely for the purpose of hardware event logging since networking is not supported under this evaluation. An example of such event logging would be hardware related disk errors. This facility is included in the TCB (as a trusted process) because of its inherent capabilities; because of these capabilities access is limited to the system administrator. DSM is bundled in with PRIMOS and is explicitly enabled through the privileged DSM_START command.

The DSM facility consists of two server processes, multiple gated procedures in ring 0, and the Inter-Server Communications (ISC) facilities. The functionality can be broken into four sections, called Remote System User (RESUS), System Information Metering (SIM), Unsolicited Message Handler (UMH), and hardware event logging.

When a user tries to use a DSM command, an ISC session is created between the user and the DSM server. The DSM server is provided the user's name, the user's current node name, and the given function the user wishes to be performed. The server checks this information in a predefined configuration file (set by the System Administrator) to determine if access to the desired command should be granted. If access is granted, the DSM server connects to the given application server which then performs the command on the user's behalf. If access is denied, an appropriate error message is returned. Access is determined by the entries in the configuration file. This file is protected by PRIMOS's ACL mechanism.

DSM Security Through Groups, Roles, and ACLs

DSM security is controlled through the mechanisms of groups, roles, and PRIMOS ACLs. There are ACLs associated with all DSM commands which provide the first level of protection against improperly using that command. DSM configuration defines groups and roles within groups which specify who may execute the various DSM/RESUS/SIM/UMH commands. A group defines all nodes which are to run under the same security policy, and a role within the group defines the ability to execute any given DSM command. By default, DSM commands for a C2 configuration will be limited to the System Administrator at any console, and the supervisor process (User 1) running at the system console.

Remote System User (RESUS)

The RESUS functionality was designed to allow a system administrator to work from any terminal in the network system by shifting the SA privileges from the system console to some other console. RESUS is explicitly enabled under DSM through the RESUS_ENABLE command. Once the system console functionality (privilege) is effectively transferred to another terminal, the only command that the system console can accept is to disable RESUS (RESUS -DISABLE -FORCE). All input and output from the System Administrator's console (wherever it may be in the system) is echoed on the system console in order that the RESUS user can be observed.

The TFM clearly states that a C2 installation should not use RESUS facilities.

System Information Metering (SIM)

The SIM facility processes fifteen (15) different commands, two of which are relevant to security in that they override an option provided in PRIMOSs ACL facilities. For a given DSM user, the LIST_PROCESSES and LIST_UNITS commands display information which includes directories even if the ACL of the directory name being displayed does not allow L(ist) access to the file.

Unsolicited Message Handler (UMH)

UMH is a DSM-controlled facility that typically filters and routes event messages in a networked environment; it can also be used in a non-networked environment by only specifying local destinations. UMH enables the System Administrator to define local (and remote in networked environments) destinations for particular types of event messages (e.g., send all disk related error messages to the terminal located by the drive units). UMH is primarily intended to service alarm and warning events from software modules and subsystems but can be used by any application that generates system management data in an asynchronous manner (e.g., diagnostic packages). UMH is similar to RESUS in the sense that it allows the System Administrator to divert console messages to different terminals but it differs from RESUS in that UMH only provides a read only message service where as RESUS provides a virtual system console.

Hardware Event Logging

PRIMOS's system event logging facility uses a mechanism that delivers all event messages to the UMH, which in turn routes these messages to DSM system logs. SYSTEM_MANAGER is the mechanism (process) that delivers event messages to UMH. By default the DSM system log is contained in the DSM*LOGS directory. This destination can be redefined through UMH.

The Spooler Subsystem

The spooler subsystem (SPOOL) is PRIMOS's file printing utility. The utility is made up of primarily ring 3 code, but all of the code that accesses the subsystem's databases is ring 0 code. This code is accessed through the spooler facility's main ring 0 gate (SP$MGR). This ring 0 gate defines the spoolers interface to its ring 0 databases. Access to this gate is restricted to the spooler itself therefore preventing illegal access to the spooler's ring 0 databases and control files.

The spooler uses the file system to store copies of the files to be printed. In particular, SPOOL_DATA is a system directory in which print jobs are spooled. This directory is ACL protected and therefore is as secure as the ACL mechanisms can make it. Some portions of the spooler (PROP, SPOOL, DESPOOL), use ring 3 shared segments to transfer status information about print jobs. This information is not security relevant (e.g., current page number, document name) so possible compromise of this information in no way violates system security.

The Batch Subsystem

The batch subsystem is a privileged process that controls PRIMOS's BATCH utility. Batch is the mechanism that allows users to execute phantom (background) jobs. By default a user's batch phantom runs at the same priority as the interactive user who created it. Users are allowed to lower their priority, but only privileged users can increase their priority. To perform the batch processing, batch is set up to use from one to sixteen queues for user submitted batch requests. These queues are manipulated by a special phantom (uid = BATCH_SERVICE) which is brought into being when the batch subsystem is started up.

All of the batch queues and control files are protected by putting them in ring 0. These can only be accessed by a set of ring 0 gated procedures. JOB$1 (one of the gated procedures) is the primary interface to the ring 0 portions of the batch subsystem. It provides all queue access on behalf of the PRIMOS users. Since it is the only mechanism that manipulates the batch queues, ordinary users cannot access the queues without first calling JOB$1.

Use of the Gate Mechanism

As described on page 20, "Hardware Protection Rings", PRIMOS is built on a three ring architecture. The most privileged ring of execution is ring 0. PRIMOS uses this ring as the primary execution domain of the TCB; all of the security critical portions of the operating system (i.e., its TCB) are contained in ring 0. The following section on the TCB will cover the software found in ring 0. The next most privileged ring is ring 1. Currently this ring is not used by PRIMOS but it is available in the hardware. Ring 3 is the least privileged ring of execution. It is here that PRIMOS users and and their untrusted software resides. Also found in Ring 3 are the system's trusted processes. Ring 2 does not exist in PRIMOS.

A critical part of the hardware ring mechanism is the mechanism which allows users to change rings of execution. PRIMOS implements a software ring crossing mechanisms with "software gates". A gate, or gated procedure, allows a user (who has access to the given gate) to access portions of the system at differing inward rings of execution. Access to a gated procedure is based on the process's access rights to the gate; a procedure's ECB must also be located in the special gate access segment. The ECBs located in this segment must all have starting addresses beginning on an ECB boundary. Since ECB entries are all of equal size, an even ECB boundary is defined as all of the locations in this segment, starting from location 0, that are 16 words apart. If a procedure accesses an improperly aligned ECB (i.e., one that does not have a starting address where the address mod 16 does not equal zero (0)), an access fault occurs. To call any procedure allowing gate accesses, the calling procedure must execute the PCL instruction that points to an ECB in the gate access segment.

The mechanism of calling a gated procedure was introduced in the section beginning on page 20, "Hardware Protection Rings", and will be explained in more detail now. Once the ring number is derived, PCL must allocate a stack for the called procedure. This is accomplished by getting a pointer to the stack root segment (offset 3 of the callee's ECB). This pointer provides the location for the new stack frame. Once the stack frame is set up, PCL saves the caller's state. The processor clears the flag field of the new frame and stores the contents of the caller's PC, stack base and link base registers, and keys into the new stack frame. The saved PC contains the ring and segment of the caller to be used by PRTN. The callee's state can now be loaded. PCL disables interrupts so that page faults, STLB faults, or interrupts cannot disrupt the system while it is loading the callee's state. The information that is loaded to set up the callee's state is taken from the callee's ECB. The final step before beginning execution of the called procedure is to compute and store the indirect pointers used in argument passing as described in the section starting on page 20, "Hardware Protection Rings". Once this is complete, execution of the called procedure begins. PRTN is used to return control to the calling procedure after execution of the called procedure is complete.

This mechanisms provides a secure means to allow inward ring procedure calls, or "gate crossing". As mentioned previously, PRIMOS will only allow inward ring crossing and outward returns. This fact is critical to the secure operation of a ring based system.

Some of the gated procedures use ISPRIV$ to provide further protection from unauthorized access. This procedure provides a means of checking whether a given user is considered privileged by PRIMOS. The procedure using ISPRIV$ only gets a boolean value returned from ISPRIV$; it is the calling procedure's responsibility to interpret ISPRIV$'s response accordingly. ISPRIV$ determines whether the caller's user type is privileged by checking if it meets certain criterion. The criterion are as follows: is the user type of the caller equal to a given (caller specified) user type, is the caller user 1, is the calling process attached to the system console, is the caller the system administrator, and/or is the caller in a specific ACL group. Access control decisions made by a user of ISPRIV$ are based on the user type of the caller by allowing the callee to specify who can use a given gated procedure and checking these user types against the caller's user type (see page 37, "TCB Protected Resources").

Parameter Checking

When a gated procedure is called by an outer ring (ring 3) process, PRIMOS will first copy the parameters from the ring 3 process stack to the ring 0 process stack, and then verify the parameters. Pointer values passed have their ring value maximized to ensure that ring 0 code or data is not read or overwritten. For I/O calls, the buffer addresses and lengths are checked against the processes address space limitations.

The Dynamic Linking Mechanism

PRIMOS uses dynamic linking of subroutines to allow more flexibility in design. These subroutines are potentially located in shared system segments; which means that any PRIMOS user can access them. Dynamic linking allows a running process to call a subroutine that had not been linked with it at bind time. There are three general types of subroutine libraries in PRIMOS: PRIMOS-resident libraries (internal), Library EPFs, and Static-mode shared libraries. The dynamic linking mechanism allows access to the entry point subroutines found in these libraries. An entry point allows further restrictions on accesses to the shared libraries by only allowing those subroutines that are entry points to be accessed by user programs.

Library EPFs can be accessed through the dynamic linking mechanism. If the EPFs have been loaded into a shared area of memory, they are available to all users. If they are not in a shared area, and the user has DAC access to the libraries on disk, the user can use the EPFs.

There is a limited search for missing entry points in a specific set of libraries (see page 34, "Search Rules"). If not found in this search, PRIMOS will return a dynamic linking error.

Dynamic linking of subroutines is accomplished by means of a dynamic link (DYNT). In place of a normal pointer to the ECB of a subroutine, a dynamic link consists of a special pointer, called a Faulted Indirect Pointer (FIP) that points to the name of the given entry point subroutine. The dynamic link serves as a place holder for a subroutine pointer until it is located and connected to the running program.

The dynamic linking mechanism comes into play when a running program attempts to use a FIP when calling an external subroutine. A procedure call (PCL) instruction whose procedure names cannot be resolved at bind time are set to point to an indirect pointer with its fault bit set. This causes a fault during the execution of this instruction, which signals PRIMOS to resolve the reference. The reference is resolved by the dynamic linking mechanism. It examines the FIP, determines that it is part of a dynamic link and gets the name of the desired subroutine. This is accomplished by "snapping" the link; in a temporary copy of the FIP, the FIPs fault bit is reset and the copied FIP is processed like an normal IP. The name of the desired subroutine is then read from the memory pointed to by the IP. Once the name is known, the list of named subroutines in a given library is searched to find the appropriate ECB address. Once the ECB address is known, the original FIP is replaced with the actual address. Because this mechanism adheres to both the software enforced file system access control and the hardware based segment control mechanisms, the dynamic linking mechanism does not circumvent system security in any way.

Search Rules

PRIMOS has a search rule facility which allows users and administrators to specify how to find different types of file objects. There are search lists to locate directories, executable files, source code files, binary objects, and EPF libraries.

Search lists contain one or more search rules, and are specified for each process. Search rules consists of three types: administrator rules, system rules, and user defined rules.

Administrator and system search rules are defined as files in the protected SEARCH_RULES* directory and may not be modified by users. In addition, a process cannot even change administrator and system rules assigned to it by PRIMOS. It is these rules which specify where directories, executable files, source code files, binary objects, and EPF libraries are found.

Users may add their own search rules on a user specified search list. It is not possible for a process to modify the search lists of another process. A subordinate process will acquire the search lists of its creator.

PRIMOS will start at the beginning of a particular search list until it finds what it is looking for or return an error.

Starting up PRIMOS

There are two phases during PRIMOS startup that lead to the full initialization (i.e., a secure state) of the system and creation of servers. User 1 accomplishes startup, and the system can be configured so that no users may log in until startup is completely finished. Two files containing start up directives are involved: the system configuration file (usually called CONFIG) and the system command input file PRIMOS.COMI. Both of these files reside in the directory CMDNC0 and their ACLs can be set to protect them against unauthorized access as specified in the Trusted Facility Manual [5].

The first phase takes place entirely in ring 0. This phase sets up the primary data structures required for execution of both system and user processes. This includes the following:

- Set the amount of physical memory available for use
- Activate system disks and the system console.
- Set up the ready list, which will subsequently be maintained by the microcode.
- Allocate the PCBs and other structures required for system processes.
- Set up the data structures for system processes, such as the DTAR values in the PCBs, the SDW entries in the SDTs, and the PMTs which are all required for the virtual memory mechanism.

- Set up the remaining scheduling queues.
- Specify the physical device numbers of the paging disks and allow the enabling of disk mirroring (see page 27, "The Copy Server").
- Download the intelligent controllers which are configured.

During this phase the configuration file (CONFIG) which contains configuration directives is read and fundamental system structures are allocated and initialized. For instance, directives, such as NTUSR and NPUSR specify the maximum number of terminal users and phantoms respectively. A specific range of Process Control Blocks (PCBs) are allocated for these users and initialized (see the section beginning on page 37, "Protected Subjects"). Another configuration directive, for example, sets the size for terminal buffers. One directive of particular interest for C2 is TPDUMP. This directive allows the taking of a crash dump immediately after PRIMOS halts unexpectedly. With the crash dump tape, an administrator can use the program CRASH_AUDIT (see page 49, "Audit"), to retrieve the auditing buffers used by the auditing mechanism. Several possible directives relate to the allocation of network processes; these directives will not be present on C2 systems. If the directive is not present, no processes are allocated for that category. NSLUSR and NRUSR, which stand for "number of slave users" and "number of remote users", are examples of these network-related directives. The final operation of the ring 0 phase is to start up LOGIN_SER[illegible] and TIMER_PROCESS.

The second phase takes place in ring 3. During this phase the command input file PRIMOS.C[illegible] is executed. Unlike the system configuration file, this file contains commands that can be issued at any time from the system console. Further initialization such as adding disks, setting the baud rate for terminal lines and starting disk mirroring can take place, but is not required. PRIMOS.COMI typically contains commands to allow sharing of programs and subsystems among users. For example, the PLP compiler is invoked through a command in CMDNC0, but most of the compiler is located in DTAR 1 and is shared for execution (i.e., READ access) among all users. This sharing reduces the system's working set. Because the segments used are always DTAR 1, they are present in every user's address space. The SHARE command sets READ access (the default) to the segments used. PRIMOS.COMI also contains commands to start server processes such as SPOOL (see page 31, "The Spooler Subsystem"), BATCH (see page 31, "The Batch Subsystem"), AUDITOR (see page 29, "The Audit Server"), and the DSM processes (see page 29, "Distributed System Management (DSM) Servers"). Only user 1 and the processes it created can run until the MAXUSR command is issued. The MAXUSR command is typically issued at the end of PRIMOS.COMI command file at a point where the system is deemed to be in a secure state, so that normal users may log in.

TCB Protected Resources

The only subjects in PRIMOS are processes. The objects in PRIMOS are files, directories, named semaphores, numbered semaphores, Master File Directories (MFDs) (also known as partitions), Inter-Process Communications (IPC), and devices. While there are many types of files (e.g., SAM files, DAM files, MIDASPLUS files, etc.) the distinction between these file types is made only by application programs. Since the TCB treats all file types simply as files or directories, the different file types will not be discussed further.

Protected Subjects

The only subjects in PRIMOS are processes. There are three types of processes: interactive, phantom, and slave. Interactive processes are processes started from a terminal and associated with a particular user. Phantom processes are processes that are started from another process and associated with the user id of the process that creates it. Slave processes are different from phantoms only in that they are used to support network activity. Since network support software is not part of the evaluated configuration, the evaluated system will have no slave processes.

Each process is represented by a Process Control Block (PCB). This data structure contains all of the information required to perform a process exchange. For example, the PCB contains the process's register save area, pointers to the segment tables for the process's private memory, the process's priority, and pointers to its stack. At system configuration time, the maximum number of active processes is established. System configuration thus creates an array of PCBs which are referenced by their index. This index is known as the process id and the PCB number. A process is assigned to a PCB based on its type and possibly on the device to which it is attached. PCBs are first divided into ranges with one set being assigned to terminal users, another to phantom processes, and still another to slave processes. For those assigned to terminal users, a particular PCB is assigned to each terminal in the system.

At system configuration time, another array is created to hold the user type of each process. The PCB number is also used to index into this array.

In each process's private address space, specifically DTAR 3, is a ring 0 segment known as segment 6000. This segment contains an area known as the user profile. This profile contains the user id, group ids, and project id of the process. Each user is associated with at least one project, however each process is only associated with one project for the life of the process. Projects are used only for accounting and resource allocation purposes. In addition, each user may be associated with zero to sixteen different groups and all processes acting on behalf of that user will always use the access rights of all groups for which the user is valid. Groups are PRIMOS's method of grouping users with similar discretionary access control requirements. Groups are only used when determining a process's access to objects.

Since each process has its own set of these data structures, multiple processes running under the same user id are distinct subjects. (For a more detailed description of these data structures and their use, see page 16, "Process Management").

Of particular interest is the user type attribute. This attribute is used to indicate the general function of the process and in many cases is used to determine the privileges that should be afforded the process. This attribute is activated at login time when the process is created and cannot be changed during the life of the process. User types are defined by a project administrator or created and defined by the system administrator by using EDIT_PROFILE. All user types are then maintained on a user profile database (For a more detailed description see page 46, "Identification and Authentication").

User types do not have a particular required associated "privilege". Instead, in each place within the TCB where a particular user type is allowed to perform some special operation, an explicit check of that user type is performed by calling UTYPE$. This procedure returns the user type of the user on whose behalf the TCB is executing. Alternatively, the TCB may call the procedure ISPRIV$. This procedure will return a boolean value indicating whether the process on whose behalf the TCB is executing is privileged (see page 32, "Use of the Gate Mechanism").

Subjects can be one of following user types:

User Type	Privileged	Description
U$LOGO		Logged out
U$LOIP		Logout in progress
U$NORM		Normal User
U$SUSR	X	Supervisor (SUSR)
U$PH		Phantom
U$CPH		CPL Phantom
U$FORK		Unix type fork phantom
U$LSR	X	Login Server
U$TMR	X	Timer Server
U$CSR	X	Copy Server
U$AUD	X	Security Auditor
U$SMSR	X	DSM System Manager Server

Each of the user types marked with an "X" in the "Privileged" column is afforded some ability to perform privileged operations based entirely on its user type. Each of the privileged user types are used as system servers that take requests from other processes and perform some function on their behalf (see page 28, "Ring 3 Trusted Processes").

The following user types are valid PRIMOS user types but they are explicitly excluded from this evaluation because they are only available when using the network software which has been excluded from the evaluated configuration (see functional testing section):

User f Type	Privileged	Description
U$TREM		User gone remote
U$FREM		User from remote
U$THRU		User logged through
U$TFAM		Old-style user FAM
U$NPX		NPX slave
U$PFAM		Old-style phantom FAM
U$NET		Network process
U$RTS		Route through server
U$NSR		User written network process

Process Creation

When a user attempts to login to PRIMOS, the login server prompts the user for a user id, password, and possibly a project id. It uses the user id and password to determine that this is a valid user. It then checks to insure that this user is authorized for the specified project. Once it has completed these checks it initializes the PCB and user type that will belong to this user's process. The process is assigned a set of segment tables to describe the private half of its address space. It then creates the user profile and fills it in with values determined by the SAD entry for this user.

Phantom processes are created by calls to the TCB. These calls create a process as described above except there is no exchange of user identification and authentication information as the phantom always uses the user id of the process that created it. Phantom processes differ from interactive processes only in their user type and their PCB number.

Protected Objects

The objects in PRIMOS are files, directories, named semaphores, numbered semaphores, MFDs (also known as partitions), IPC, and devices. Each of the following subsections briefly describes one of these object types, the modes of access allowed, and the conditions necessary to exercise each access mode.

In each of the following descriptions the "effective discretionary access mode" refers to the result of the discretionary access control check as described on page 53, "Determining Access". Access modes not specified for a particular object type have no effect on the access calculation.

Files

Files are the primary information containers for the system. The contents of files are not controlled by the TCB in any way.

read (R)	A process granted read access to a file may open the file for read. A process may exercise "read" access if and only if the effective discretionary access mode includes "R".
write (W)	A process granted write access to a file may open the file for write. A process may exercise "write" access if and only if the effective discretionary access mode includes "W".
execute (X)	A process granted execute access to a file may cause the file's EPF to be mapped into his address space. (This operation is only allowed for dynamic program files.) A process may exercise "execute" access if and only if the effective discretionary access mode includes "R" or "X".

owner (O)	A process granted owner access to a file may alter the access permissions for the file and alter the read/write lock[1] for the file. A process may exercise "owner" access if and only if the effective discretionary access mode includes "O".
none (NONE)	A process granted NONE access to a file cannot access the file in any way.

Directories

Directories are file system objects that are controlled by the TCB. The primary purpose of directories is to provide a repository for the location of those file system objects immediately subordinate. There are two types of directories, user file directories and segment directories. These types of directories are the same in every respect except that user file directories contain files that can be referenced by an alphanumeric identifier chosen by the file creator while segment directories contain files that can only be referenced by the ordinal number of their directory entry.

use (U)	A process granted use access to a directory may attach to the directory (make it the current working directory). A process may exercise "use" access if and only if the effective discretionary access mode includes "U".
list (L)	A process granted list access to a directory may list the contents of the directories (i.e., list the names of the objects in the directory). A process may exercise "list" access if and only if the effective discretionary access mode includes "L".

1 See page 46, "Read/Write Concurrency Locks".

add (A)	A process granted add access to a directory may create new entries in a directory. A process may exercise "add" access if and only if the effective discretionary access mode includes "A".
delete (D)	A process granted delete access to a directory may delete existing entries from the directory. A process may exercise "delete" access if and only if the effective discretionary access mode includes "D".
protect (P)	A process granted protect access to a directory may alter any access right of the directory or any access right of any object in the directory. In addition, a process granted protect access to a directory may alter the read/write lock[1] of the directory or any object in the directory. A process may exercise "protect" access if and only if the effective discretionary access mode includes "P".
owner (O)	A process granted owner access to a directory may alter any access right of the directory (except a protect access right) or any access right (except a protect access right) of any object in the directory. In addition, a process granted owner access to a directory may alter the read/write lock[2] of the directory

1 See page 46, "Read/Write Concurrency Locks".

2 See page 46, "Read/Write Concurrency Locks".

	or any object in the directory. A process may exercise "owner" access if and only if the effective discretionary access mode includes "O".
none (NONE)	A process granted NONE access to a directory cannot access the directory in any way.

Devices

Devices are the external I/O devices of the system which the user may access directly. For example, tape drives and private disk drives. It does not include devices used solely by the system, for example, disk drives with a file system mounted or terminals. Devices are represented in the file system by entries in the DEVICE* directory.

use (U)	A process granted use access to a device may use the device. A process may exercise "use" access if and only if the effective discretionary access mode includes "U".
none (NONE)	A process granted NONE access to a device cannot access the device in any way.

DAC On Tapes

PRIMOS provides a method where several users may use a tape drive one at a time. Access to the tape drive is protected by an ACL which is initially set to those users who are allowed access. A ring 0 gated procedure in PRIMOS can be called to assign the tape drive to one specific individual since it will change the tape drive ACL so that it is held exclusively by the first user requesting it. The first call to this procedure will merely change the tape drive ACL for exclusive use. A second call to this gated procedure, using the "MOUNT" option will send a message to the system console and suspend the calling process. The message sent to the system console will contain the following information:

1. a request to mount a tape reel,
2. a specified tape drive (the one already assigned),
3. the name of the user, and
4. an identifier which may be found on the tape reel label.

The paper label on the tape reel contains the tape reel identifier and the list of users who are allowed to access this tape reel. The operator will then locate the tape reel and determine if the requesting user is allowed access.

If the user is allowed access, the operator loads the tape reel on the required drive, and acknowledges the "MOUNT" request. At this point, the user process is allowed to proceed with the use of the tape drive.

If the user is not allowed access, the tape drive is not loaded by the operator, and the "MOUNT" request is denied and the user process continues without getting access to the tape reel.

In either case, it is the responsibility of the user to unassign the tape drive thereby resetting the ACL from exclusive use to the original tape drive ACL. Note that the only way the user can do this is through an UNASSIGN gated procedure call. The ACL of the DEVICE* directory is set such that only a very privileged user can actually change the the ACL.

This DAC mechanism provides control over the tape drive only. In essence, a user who is authorized to use the tape drive has access to any tapes which the operator may mount on the drive. The operator is the individual which provides the DAC mechanism for the tape reels themselves, by deciding whether a user should be allowed access to any given tape reel depending what is on the paper label on the tape reel.

Named Semaphores

Named semaphores are objects that are designed to be used to implement process synchronization. Named semaphores are part of the file system and are treated like files for access control.

read (R)	A process granted read access to a named semaphore may observe the current state of the semaphore. A process may exercise "read" access if and only if the effective discretionary access mode includes "R".
write (W)	A process granted write access to a named semaphore may alter the value of the semaphore. A process may exercise "write" access if and only if the effective discretionary access mode includes "W".

owner (O)	A process granted owner access to a named semaphore may alter the access permissions for the named semaphore. In addition, a process granted owner access to a named semaphore may alter the read/write lock[1] of the semaphore. A process may exercise "owner" access if and only if the effective discretionary access mode includes "O".
NONE (none)	A process granted NONE access to a named semaphore cannot access the named semaphore in any way.

Numbered Semaphores

Numbered semaphores are objects that are designed to be used for process synchronization. The system has 64 numbered semaphores which are available to any process at any time. Each numbered semaphore is ACL protected. This is accomplished by having each of the numbered semaphores represented as ACL protected file system objects (i.e., directories). When a numbered semaphore is accessed, the access control decision is based on the ACL of the directory representing the given numbered semaphore. The only right that is applied to numbered semaphores is USE ('U'). All other rights are ignored.

Master File Directories

Master File Directories (MFD) or partitions as they were previously called, are the file system representation for the logical volumes. MFDs are exactly like directories except that they contain the additional information necessary to manage the logical volume (e.g., file allocation map). Access control for MFDs is exactly the same as that described for directories on page 41, "Directories".

1 See page 46, "Read/Write Concurrency Locks".

Inter-Process Communication (IPC)

PRIMOS supports an ACL protected inter-process communication mechanism through mailboxes whose ID is unique for a given system. Mailboxes can be viewed as virtual devices, allowing users to read and write messages to them. Access to a mailbox is determined by an access category whose name is the same as the mailbox, and which lists all users authorized to use that particular mailbox.

The IPC mechanism is supported through gated procedures in ring 0, and allows users to open, close, delete mailboxes, and send and receive messages. A user may send a message to a specific user, or send a message to all users (who have access to the mailbox). A user may also receive a message from a specific mailbox, or any mailbox and has the option of suspending his/her process until a message arrives.

Any user with access to a mailbox may also get a list of all users currently using that mailbox, and receive IPC status information. IPC status includes such information as the number of mail messages for this user in a mail box, maximum amount of space left in a mail box, and the number of users attached to the specified mailbox.

Since the IPC mechanism is file based, the opening and closing of IPC sessions is audited.

Read/Write Concurrency Locks

An attribute of certain objects is that they have a read/write concurrency lock which determines how many readers and/or writers may use it concurrently. The lock may be set for only one reader, one writer, one writer and multiple readers, or several readers and writers. The implementation of Read/Write locks, makes the lock part of the directory entry for the object. A user needs PROTECT ('P') or OWNER ('O') rights to change the RWLOCK of an object.

TCB Protection Mechanisms

PRIMOS provides four primary protection mechanisms: identification and authentication, audit, discretionary access control, and object reuse. These mechanisms are described in detail in the following subsections.

Identification and Authentication

In order to log in to a PRIMOS system, all users must first get a user id, password and an optional project id from the System Administrator. A user id has up to 32 characters where the first character must be a letter. Passwords may have up to 16 characters and are stored in encrypted form by PRIMOS. The CHANGE_PASSWORD command allc 's users to change passwords.

The LOGIN SERVER is a PRIMOS process which is initialized at system start-up and monitors all inactive lines for activity. The LOGIN SERVER possesses the capability to require that each user provide a user id, password, and project id before being allowed to access the system. The information presented by the user is checked against information contained in the user profile data base (which is contained in the SAD) to ensure that the user is authorized to access the syster . . e System Administrator is the on'y user who can add entries to the user profile data base and the[illegible]e, is the only user that can add new users to the system.

The information for project and user profiles is created and maint·ined by the privileged utility EDIT_PROFILE and placed in various files within the System Administrator Directory (SAD). These files are the User Validation File, Master Project File, Master Group File and System Default File. The default ACL for the SAD allows access only to the System Administrator (i.e., the ACL has the form SYS_ADMIN: ALL, $REST: NONE). The default ACLs of the files within the SAD will be identical to those of the SAD unless set otherwise.

User and project administration and maintenance is performed by system and project administrators through the use of the EDIT_PROFILE utility. EDIT_PROFILE is a privileged utility that can only be run by a system or project administrator or by any user at the system console. EDIT_PROFILE executes in one of three modes:

1. initialization mode
2. system administration mode, and,
3. project administration mode.

New users are defined to PRIMOS through the ADD_USER command in EDIT_PROFILE. Project administrators can also use this command to add already existing users to already existing projects that th[illegible] administer. New projects are defined to PRIMOS through the ADD_PROJECT command. Use of this command is limited to system administrators only. If a SAD does not exist, users can not log in to PRIMOS. The SAD may only be created from the system console by the User 1, named "SYSTEM", while EDIT_PROFILE is in Initialization Mode.

If users provide valid log in informatic.. to the LOGIN SERVER, users are attached to their origin directory (specified in the User Profile Data Base), their system and project access rights are set, and an audit record may be generated if this type of event is being audited.

If a user fails to provide valid log in information, the log in attempt is denied and an audit record and/or system console message is generated depending upon whether auditing and alarm messages have been turned on.

PRIMOS provides the system administrator with several options that allow for varying degrees of log in security. Some of these options are necessary for PRIMOS to satisfy the C2 requirements for Identification and Authentication. The options available to the System Administrator are:

NO_NULL_PASSWORD	Either allows or prohibits the use of null passwords in PRIMOS. When this option is turned on (to prohibit null passwords), EDIT_PROFILE produces a list of users who currently have null passwords so that the system administrator can assign passwords to them.
MINIMUM_PASSWORD_LENGTH	Allows the system administrator to set a minimum length for user passwords. The minimum length can be set from 0 to 16 characters. When used to assign a non-zero length, existing passwords are not affected.
FORCE_PASSWORD	Prevents PRIMOS from accepting passwcrds entered on the same line as the LOGIN command. Forces users to enter their password after the "Password?" prompt is issued.

The Trusted Facility Manual [5] instructs the system administrator to disallow null passwords and to require that passwords be entered on a separate line from the LOGIN command so that they will not be displayed. The TFM also recommends that the system administrator set minimum password length equal to six characters.

A password is not displayed when it is entered unless it is entered on the LOGIN line, or the user is on a half-duplex line. A password is always displayed on a half-duplex line. The FORCE_PASSWORD command should alays be used in a C2 configuration.

The operator never logs in at the system console or at any other console under PRIMOS. The process which executes at the system console at system startup time is User 1 and is a very privileged user.

In general, the LOGIN SERVER has no interface with users already logged in and its command repertoire includes only DATE, DELAY, DROPDTR, LOGIN, and USRASR.

The DATE command simply provides the date, the DROPDTR command disconnects a modem line, and the DELAY command defines a time function which delays the printing of a character after a carriage return has been output to a terminal (used on terminals with unusual line speeds). The USRASR command allows the System Administrator to place the system console in user mode and lock it in this mode. This allows a system with a small number of terminals to use the system console as a user terminal. All messages sent to the system console while it is in user mode are buffered until the time that the System Administrator places it back in system console mode through the specification of a password.

Every process executing under PRIMOS has a unique process identification number. Every audit record contains both the user name along with the process id which generated that specific auditable event.

There are several system configuration directives which affect the log in and log off process. A user who is already logged in to PRIMOS and wishes to log in again (second process level) may be forced to log out first depending on the setting of the LOGLOG directive. The DISLOG directive enables or disables automatic log out for disconnected lines. The LOGMSG directive enables or disables the display of log in and log out messages on the system console. The LOTLIM directive specifies the log in time limit (default is 3 minutes) before a log in attempt fails. The LOUTQM directive specifies the length of time with no activity that is allowed before automatic log out is invoked.

Audit

The PRIMOS security audit facility has four components; an audit collection facility, an audit reporting facility, an audit file backup facility, and a crash audit recovery facility.

The audit collection facility runs as a separate system process (AUDITOR) that can be started and stopped by the System Administrator from any terminal or by any user from the system console. The audit collection process can monitor any or all users, logging all their events or only a specified set of events. The facility also provides the capability of selecting only those events with a specific result (e.g., all successful attaches or all file system events which fail because of insufficient access authority).

Events are categorized into four groups: file system events, attach events, privileged operation events, and system events. File system events are actions involving creating, opening, closing and deleting of files (also includes IPC) and ACL modifications. Attach events are the actions of moving around within the file system. Privileged operation events are actions which require special privileges such as commands restricted to the system console or to privileged users. System events are actions involving normal log ins and log outs, phantom log ins and log outs, assignment of devices, segment allocation and deallocation and program mapping and unmapping. The Trusted Facility Manual [5] lists all events capable of being audited.

PRIMOS provides the System Administrator with the ability to manage the audit system through the SECURITY_MONITOR command. The audit system is activated by the

SECURITY_MONITOR -START command, The TFM [5] recommends that this be done during system start-up. The SECURITY_MONITOR command allows the System Administrator to enable and disable auditing of specific users, groups of users, events, and event results. It also allows the System Administrator to specify the audit file through the -OUTFILE control argument. If no specific pathname is given with this argument, the file will be created as SECURITY_LOG.YYMMDD.HHMMSS in the current directory.

The Audit Reporting facility provides the SECURITY_STATUS and PRINT_SECURITY_LOG commands. The SECURITY_STATUS command provides the System Administrator with a brief report on the status of the audit collection facility, such as which users and events are being monitored or the audit file pathname. The PRINT_SECURITY_LOG command allows the System Administrator to invoke an audit file reduction tool to narrow the view of the audit file to obtain information on certain users or event categories.

The PRIMOS audit facility also provides the capability of saving audit files to tape or disk using the TRANSFER_LOG utility.

The last utility provided by the PRIMOS audit facility allows the recovery of audit records in a PRIMOS system image dumped to magnetic tape after a system crash. The CRASH_AUDIT utility removes the audit records from the system image dump file and places them in sequence, to the audit file specified. This utility requires that the system configuration directive TPDUMP -YES be given in order to provide a tape dump of the system on a system crash, and a ring 0 load map file be available for the system image.

An audit file full condition will temporarily suspend all processing under PRIMOS and place the system console in a mini-command environment where the only possible commands available are to switch the audit files and other innoculous commands. Once the system administrator has rectified the problem, processing continues normally. If the system console is in user mode at the time processing is suspended, the system administrator will be unable to get back to the system console. PRIMOS will halt (crash) before it is forced to lose any audit records.

Shared segments are allowed only for system and wired class storage and are accessible only to ring 0, so users never have free access to this information. Segments used for user class storage are private and only old information belonging to the current user is accessible.

Discretionary Access Control

To use the PRIMOS file system, a user must first be attached to a file directory. A file directory is the file that contains the names and locations of other files on the given disk partition. An attach point is a point of reference in the PRIMOS file system's tree structure that is used as the current working directory for a given user.

To attach to a node of this tree a given user would use the ATTACH command. An attach operation is limited by the fact that a user must have the proper access rights to the directory he wishes to attach to. At the system level, the attaching process is implemented by a set of ring 0 gated procedures that provide the capability to attach to any specified attach point.

This set of gated procedures is used to support the low-level file system operations by providing a mechanism to attach to, and move around in, the PRIMOS file system. The correct operation of this set of gates is an essential step in PRIMOS's manipulation of files on disks because all of the other file system related functions depend on these gates to work properly. Access control decisions are made by these gates through the use of the ACL-related gates.

To manipulate any objects in the file system, a user must be attached to some directory in the file system. Since this initial attach to the file system uses the ACL-related gates to ensure that access to this point in the file system is allowed, the first step in manipulation of file system objects is governed by Discretionary Access Control.

The ACL mechanism is implemented by a set of ring 0 gates. These gates provide PRIMOS users with a means of protecting their file system objects with ACLs. An ACL provides a way of limiting access to a given object by named individuals and/or groups of individuals; this mechanism is the realization of the C2 requirement of Discretionary Access Control (DAC). The user visible facilities used in the manipulation of the ACLs are implemented by using the following ring 0 gates:

AC$CAT0:	Add an object's name to an access category.
AC$DFT0:	Set an object's ACL to that of its parent directory.
AC$LIK:	Set an objects ACL to that of another object.
AC$LST0:	Obtain the contents of an object's ACL.
AC$RVT:	Convert an object from ACL protection to password protection.
AC$SET0:	Set a specific ACL on an object.
CALAC$0:	Determine whether an object is accessible.
CAT$DL0:	Delete an access category.

GETID$:	Get the user ID and the groups to which it belongs.
ISACL$:	Determine if an object is ACL protected.
PA$DEL:	Remove an object's priority ACL.
PA$LSTO:	Obtain the contents of an object's priority ACL.
PA$SET:	Set priority access on an object.

This set of gates provides the low-level interface to the ACL mechanism for the user. They ensure the proper handling of ACL related functions.

There are two types of Access Control Lists (ACLs) in PRIMOS: Specific and Named. Only one type may be applied to an object at any time.

Specific ACLs

Specific ACLs exist as an attribute of the object that it protects and are implemented as part of the directory entry for that object. They do not appear in the file system as distinct objects. When the object that the Specific ACL protects is deleted, the ACL is deleted as well. There are three types of Specific ACLs: Standard, Priority and Device.

Standard ACLs are used to grant or deny access to file system objects (e.g., directories and files) and can be manipulated by the "owner" of the object. A subject "owns" an object when the subject has the PROTECT access mode to the directory in which the object resides.

Priority ACLs are used to grant or deny access to entire disk partitions and can only be set by a System Administrator or System Operator from the system console or supervisor terminal. Priority ACLs override the access mode settings of Standard ACLs.

Priority ACLs are usually used only for special operations such as backups. This is done by setting a Priority ACL for SysDaemon to the entire disk partition.

Device ACLs are used to grant or deny access to system devices such as tape and disk drives. Device ACLs can only be set by a System Administrator or System Operator from the system console or supervisor terminal.

Named ACLs

Named ACLs exist as distinct file system objects called access categories. As such they can be used to protect more than one object and they themselves are protected by their containing directory's ACL (i.e., access to an access category is mediated in the same way as access to a file). Access categories must reside in the same directory as the objects that they are protecting and their names must end in the suffix ".ACAT". An access category contains an ACL as its contents, this ACL is used for Discretionary Access Control decisions when attempts are made to access the protected object. If an access category that is currently being used to protect an object is deleted, the protection for that object reverts to the default protection provided by a parent directory.

Determining Access

Access to an object in PRIMOS is determined by checking for the requesting user on the ACL for the requested object. If a user is specified by name on the ACL for an object, the user will be given only those access rights associated with their name. If a user belongs to one or more groups specified on the ACL for an object as well as being specified by name, the user will be given only those access rights associated with their name on the ACL. If a user belongs to one or more groups specified on the ACL, and they are not specified by name, the user will be given the sum of the access rights given to each of those groups (when calculating the sum of the access permissions, the "none" access permission does not deny other permissions that may have been given to other groups). If a user is not a member of any of the groups listed on the ACL and the user is not specifically listed by name, the user will be given only those access rights that are associated with the special identifier "$REST". $REST is a special identifier that signifies those access rights that are to be given to all users not otherwise specified on the ACL.

Default Access

Access permission to newly-created objects is, by default, set to the permissions of the containing directory. If no specific permissions have been set on the containing directory, the access permissions of the next highest directory in the file system structure are used. This default access scheme allows each user to control the default access permissions for the objects they create if the System Administrator grants Protect (P) or Owner (O) access permission to the user for their own origin directory.

It should be noted that the protections applicable to directories and protections applicable to files are not the same. Therefore, for default access to work correctly, a user must specify file access modes on directories. These invalid directory modes do not affect the access calculations on the directory itself but are only used when needed to calculate access for an unprotected subordinate object.

Access Revocation

Revoking access to an object from a user or group that previously had access permission can be done in either of two ways. The first is to delete that user or group name from the access list. This will still allow these users to access the resource if access permission has been set for $REST. The second way is to re-specify access for the user or group with NONE access permission. This will deny access even if a $REST permission has been set. Access calculations only occur when the user requests that an object be opened. Therefore, new access permissions do not take effect if a user already has the object open.

Object Reuse

There are three areas of concern in PRIMOS related to object reuse. They are process registers, memory space and mass storage space. The handling of each of these areas is discussed in the following sections.

Process Registers

The object reuse concern here is related to the initial contents of the registers when the user gets control of the CPU for the first time. After that time, all registers are restored with the contents that existed at the time the process was removed from control.

In PRIMOS the user is attached to its associated process following the validation of that user. All work done by the user, or on behalf of the user, is executed using the user's process. Completion of the setup of the initial user state is accomplished with this user process. Thus the state of the user process's registers at the time the user can get control represents the state of the system service that turned control over to the user (TCB code). Based on this system design there is no object reuse concern related to the user registers.

The validation process where user ids and passwords from the SAD may be found in the registers is executed using a system process. Control is not given to the user process until this validation is complete.

Memory Space

PRIMOS manages memory in segments and pages over the life of the task. Once a segment is allocated that segment remains for the life of the task. Space is allocated and deallocated within a segment through the use of a set of system calls (depending on the type of space desired). References to this allocated area are handled by the standard paging mechanism. All page faults under PRIMOS are serviced by the routine PAGTUR. If the page exists on the paging device (space previously used by the process), then the page record will be read in destroying any old information in the page frame. If the page does not exist on the paging device (newly allocated space in the segment), the page frame will be be zeroed out by the procedure FILPAG.

The reliability of this mechanism is based on the proper setting of the "no copy on disk" bit for each page. These bits are correctly set by the procedures VINIT$ and GETSEG which are the only two procedures which are responsible for setting up page maps when segments are created for a user.

Shared sesgments are allowed only for system ane wired class storage and are accessible only to ring 0, so users never have free access to this information. Segments used for user class storage are private and only old information belonging to the current user is accessible..

Thus on initial allocation to the process, memory is cleared. Reuse on dynamic allocation and deallocation within the task is up to the task. Since the TCB uses different segments which are part of the users address space but are not accessible to the user, there is no migration of data through the TCB code that runs with the user process.

Physical Disk Records

PRIMOS only allows for logical contiguous file space to be allocated, and requires that any area of the file must be written before it can be read. The system calls that extend files and perform I/O will not allow the file pointer to move beyond the physical end-of-file, nor allow a file block to be written at other than within the current bounds of the file, or across the end-of-file (this action does cause a change in the physical end-of-file). Thus all files are allocated in a sequential fashion regardless of the access method used (e.g., random, sequential, indexed sequential, etc.).

The three procedures whose concern it is to extend a file system object: ADD_REC (extends files), ALC_REC (extends directories), and NEWDAM (extends DAM files), all call the procedure GETREC which is the only procedure which allocates disk records.

In each case, GETREC is called with a key indicating that there is no copy of the new record on the disk (i.e., the record is empty). When GETREC is called with this key, it does not attempt to read the record from the disk, but simply provides a blank buffer. The old information on the disk is destroyed when the new record is written.

CAM files (a contiguous allocation file type) extend themselves by moving the logical end-of-file. The physical end-of-file remains where it was and data between the physical and logical ends of the file cannot be referenced.

EVALUATION AS A C2 SYSTEM

Discretionary Access Control

Requirement

> The TCB shall define and control access between named users and named objects (e.g., files and programs) in the ADP system. The enforcement mechanism (e.g., self/group/public controls, access control lists) shall allow users to specify and control sharing of those objects by named individuals, or defined groups of individuals, or by both, and shall provide controls to limit propagation of access rights. The discretionary access control mechanism shall, either by explicit user action or by default, provide that objects are protected from unauthorized access. These access controls shall be capable of including or excluding access to the granularity of a single user. Access permission to an object by users not already possessing access permission shall only be assigned by authorized users.

Applicable Features

PRIMOS defines and controls access between users and objects through access control lists that allow access to be specified to individual users and/or defined groups of users. Only authorized users (e.g., users with PROTECT or OWNER permission to the directory containing the object or OWNER permission to the object itself) can grant or revoke access permissions to an object.

Default access control for objects is determined by the access control permissions of a parent directory. In order for this default access control mechanism to prevent unauthorized access, a system administrator must correctly set the access permissions on the origin directory for each user.

Through the PRIMOS group mechanism, access can be granted or denied to an entire group of named individual users. Access control can be further restricted or expanded by using individual user ids.

A complete description of the DAC mechanism is provided on page 40, "Protected Objects", and on page 50, "Discretionary Access Control".

Conclusion

PRIMOS satisfies the C2 Discretionary Access Control requirement.

Additional Requirement (B3)

The following changes are made in this requirement at the B3 level:

> CHANGE: The enforcement mechanism (e.g., access control lists) shall allow users to specify and control sharing of those objects, and shall provide controls to limit propagation of access rights. These access controls shall be capable of specifying, for each named object, a list of named individuals and a list of groups of named individuals with their respective modes of access to that object.
>
> ADD: Furthermore, for each such named object, it shall be possible to specify a list of named individuals and a list of groups of named individuals for which no access to the object is to be given.

Conclusion

PRIMOS satisfies[1] the B3 Discretionary Access Control requirement. PRIMOS access control lists can include both users and groups, each of which can either be specifically granted or denied access.

<u>Object Reuse</u>

Requirement

> All authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation, or reallocation to a subject from the TCB's pool of unused storage objects. No information, including encrypted representations of information, produced by a prior subject's actions is to be available to any subject that obtains access to an object that has been released back to the system.

Applicable Features

As described on page 54, "Object Reuse", PRIMOS provides object reuse for registers, memory and mass storage space. Process register reuse is provided by the scheduling and dispatching portions of PRIMOS. PRIMOS's implementation of paged, virtual memory provides object reuse protection

1 Although PRIMOS satisfies this requirement at the B3 level, it does not satisfy the assurance requirements above its rated level.

for objects in memory. PRIMOS's file system management routines provide object reuse protection for objects in the file system.

The TFM [5] provides guidelines on how to protect off-line media and on how to safeguard accesses to devices themselves, such as tape drives, disk drives, and printers.

Conclusion

PRIMOS satisfies the C2 Object Reuse requirement.

Identification and Authentication

Requirement

> The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. Furthermore, the TCB shall use a protected mechanism (e.g., passwords) to authenticate the user's identity. The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user. The TCB shall be able to enforce individual accountability by providing the capability to uniquely identify each individual ADP system user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual.

Applicable Features

All users of PRIMOS, except the operator, are required to have a username and password to login under PRIMOS. System administrators may optionally require a project name. The user profile data base is kept in the System Administrator's Directory (SAD) and is ACL protected. Every process executing under PRIMOS is uniquely identified through a a process ID number. This process ID number and a username is provided in every audit record generated by that process. A more complete discussion of this topic is covered on page 46, "Identification and Authentication".

Conclusion

PRIMOS satisfies the C2 Identification and Authentication requirement.

Audit

Requirement

> The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data. The TCB shall be able to record the following types of events: use of identification and authentication mechanisms, introduction of objects into a user's address space (e.g., file open, program initiation), deletion of objects, actions taken by computer operators and system administrators and/or system security officers, and other security relevant events. For each recorded event, the audit record shall identify: date and time of the event, user, type of event, and success or failure of the event. For identification/authentication events the origin of request (e.g., terminal ID) shall be included in the audit record. For events that introduce an object into a user's address space and for object deletion events the audit record shall include the name of the object. The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity.

Applicable Features

The PRIMOS Security Audit facility is sold as a separate product and has four components: an audit collection facility, an audit reporting facility, an audit file backup facility, and a crash audit recovery facility.

The audit collection facility runs as a separate system process (AUDITOR), and can monitor all users or a specified set of users, logging all events, or a specified set of events. The facility even provides the capability of selecting those events which complete in a desired way, such as all successful attaches or all file system events which failed due to lack of access rights.

Events are categorized in four groups: file system events, attach events, privileged operation events, and system events (see page 49, "Audit"). The TFM lists all the specific events being audited and the addendum to the TFM displays the format of all the possible audit records and events.

If no pathname is specified, the default audit file will be SECURITY_LOG.YYMMDD.HHMMSS in the current directory. The default ACL for the audit file will be determined by the ACLs of the directory in which it resides.

Every audit record contains the following fields; a time and date stamp, the name of the user, the process id, the event type, an error status (success/failure indicator), and other relevant information depending on the event. The other relevant information may include the terminal id for identification and authentication events or the name of the object for events concerning the introduction of an object into an address space or the deletion of an object from an address space.

The PRINT_SECURITY_LOG command allows the System Administrator to invoke an audit file reduction tool to narrow the view of the audit file from the perspective of one or more users or event categories. The PRINT_SECURITY_LOG utility allows for the inspection of any audit file including the one currently being written to.

The Security Audit facility can be disabled from auditing all events and still remain a running process so users will see the AUDITOR phantom as a system process. This might discourage users from attempting to violate system security rules while the audit facility was not running at all.

Switching to another audit file without first disabling the multi-user environment may result in the loss of certain file name and unit connectivity information between the two audit files.

Conclusion

PRIMOS satisfies the C2 Audit requirement.

<u>System Architecture</u>

Requirement

> The TCB shall maintain a domain for its own execution that protects it from external interference or tampering (e.g., by modification of its code or data structures). Resources controlled by the TCB may be a defined subset of the subjects and objects in the ADP system. The TCB shall isolate the resources to be protected so that they are subject to the access control and auditing requirements.

Applicable Features

The software TCB for PRIMOS is maintained in its own domain of execution by having all of its kernel code, data structures, and some system processes entirely contained in ring 0 and by having the trusted processes in ring 3 in their own address space. Since the PRIMOS kernel is made up of ring 0 code, and since the only way to access the PRIMOS kernel from outside of ring 0 is through the trusted gate crossing mechanism, as described on page 32, "Use of the Gate Mechanism", or messages to servers, PRIMOS is effectively isolated from external interference or tampering. All processors on which PRIMOS runs provide both privileged (i.e., accessible by ring 0 processes) and unprivileged instructions as described on page 23, "Privileged CPU Instructions", that allow PRIMOS to further protect system critical operations (e.g., process exchange, disk I/O) from unauthorized access.

PRIMOS defines a set of subjects and objects whose interactions are controlled by the TCB as described on page 37, "TCB Protected Resources". Processes are the only type of subjects in PRIMOS. Process isolation is guaranteed by PRIMOS's use of PCBs (which allows isolation of the data structure that describes a process), the use of a per process private address space (which provides users with isolation of data and executables), the use of a per process private stack space, and TCB controlled shared address space (which provide users an area to share such things as common routines). Subjects are further protected by the ring mechanism as described on page 32, "Use of the Gate Mechanism", and virtual segments as described on page 19, "Virtual Memory Segments".

Objects (i.e., files, directories, named semaphores, numbered semaphores, MFD, and devices) are protected by the ACL mechanism because they are file system objects.

Conclusion

PRIMOS satisfies the C2 System Architecture requirement.

<u>System Integrity</u>

Requirement

> Hardware and/or software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB.

Applicable Features

All Prime systems being considered for this evaluation have diagnostic software embedded on-board in ROM which is executed each time the physical machine is powered on. All PRIMOS systems are also provided with additional diagnostic software which can be used to exercise hardware components in order to isolate hardware problems.

The Diagnostic Test System (DTS) is a set of architectural diagnostics which are used for testing Prime 50 Series CPU's, controllers, and peripherals. These diagnostics validate hardware and microcode, and detect hardware failures. DTS consists of two diagnostic monitors, the first of which executes in a stand-alone environment (SAM), and the second executes under PRIMOS. The stand-alone version requires a minimum hardware configuration of a 50 Series CPU, 128 KBytes of memory, a 300 baud CRT on address 4, and a boot device (tape, disk or PAL). A more complete description of the DTS may be found in Prime's Diagnostic Test System PET3.

Conclusion

PRIMOS satisfies the C2 System Integrity requirement.

Security Testing

Requirement

> The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. Testing shall be done to assure that there are no obvious ways for an unauthorized user to bypass or otherwise defeat the security protection mechanisms of the TCB. Testing shall also include a search for obvious flaws that would allow violation of resource isolation, or that would permit unauthorized access to the audit or authentication data.

Applicable Features

This requirement refers to the security testing performed by the formal evaluation team. Refer to (see page 69, "Functional Testing") for a description of the testing the team accomplished and the outcome of that testing.

In addition the team observed an independent run of the complete security test package for PRIMOS. This session consisted of the test machine being booted with the evaluated system and the tests being performed in two stages. The first stage was the initiation of the automated test suite. In this stage, a test driver initiates the various predefined test programs and monitors the results against a data base of expected results.

The output produced by these tests was captured and printed. The team monitored the test run and examined the output from this run.

The second stage was the execution of a series of manual tests. These tests, by their nature, were not conducive to automation. The tests were run individually based on predefined scripts. The team observed as the various test scripts were performed. The output from these tests were captured and examined by the team.

During the entire session all tests functioned as advertised.

Conclusion

PRIMOS satisfies the C2 Security Testing requirement.

Security Features User's Guide

Requirement

> A single summary, chapter, or manual in user documentation shall describe the protection mechanisms provided by the TCB, guidelines on their use, and how they interact with one another.

Applicable Features

Chapter 1 of the Security Features User's Guide [4] provides an overview of the Identification and Authentication (I&A), Discretionary Access Control (DAC), and Audit mechanisms. Chapters 2-4 provide detailed descriptions, as well as, guidance and examples on the use of the I&A and DAC mechanisms.

Conclusion

PRIMOS satisfies the C2 Security Features User's Guide requirement.

Trusted Facility Manual

Requirement

> A manual addressed to the ADP system administrator shall present cautions about functions and privileges that should be controlled when running a secure facilit. The procedures for examining and maintaining the audit files as well as the detailea audit record structure for each type of audit event shall be given.

Applicable Features

The manual entitled System Administrator's Guide, Volume III: System Access and Security is Prime's Trusted Facility Manual (TFM) [5]. Cautions about functions and privileges that should be controlled when running a secure facility are clearly explained over its eleven chapters. Chapter 11 explains the procedures for examining and maintaining the audit files and the addendum to the TFM provides the detailed audit record structure for each type of audit event.

The TFM covers the following topics:

Chapter	1:	Issues Affecting System Access and Security
Chapter	2:	Equipment and Environment
Chapter	3:	Backups
Chapter	4:	Planning the User Environment
Chapter	5:	Setting System Access
Chapter	6:	Using EDIT_PROFILE
Chapter	7:	Security
Chapter	8:	Adding Subsystems
Chapter	9:	Looking After Users
Chapter	10:	System Monitoring
Chapter	11:	Security Audits

Conclusion

PRIMOS satisfies the C2 Trusted Facility Manual requirement.

Test Documentation

Requirement

> The system developer shall provide to the evaluators a document that describes the test plan, test procedures that show how the security mechanisms were tested, and results of the security mechanisms' functional testing.

Applicable Features

Prime provided a document entitled "Test Plan for C2 Security". This document described the security policy and presented an overview of the areas of the system that were to be tested. The document provided a description of the testing philosophy and the organization of the tests and the testing methodology. Since this methodology is unique, it is described in detail below.

PRIMOS is a ring-structured system in which the operating system is separated from the user by the ring mechanism. Security functional testing is done at the gate level as this constitutes the user interface to the TCB.

Prime chose a testing method that provides complete coverage of the TCB interface without testing every gate's function extensively. This has been done by placing every gate in one of three categories. A description of the three categories, the criteria for gate selection and the testing requirements for those categories are presented below.

1. Those gates that implement the primary user interface to a security-relevant function (e.g., a gate which modifies the contents of an ACL).

   This category contains those gates that reference a routine that:

   - makes an access control decision
   - manipulates data that will be used in an access control decision
   - mediates access to a global table of non-public information.

   These gates are tested to show functional correctness and that they abide by the security policy.

2. Those gates that implement a user interface to a function that is expected to call a category 1 gate before performing its specified function.

   This category includes those gates that reference routines that call a category 1 gate to make an access control decision but could ignore the results of that call.

   Category 2 gates are tested to show they do in fact call the appropriate category 1 gate and abide by its decision.

3. Those gates not falling into categories 1 or 2 above (e.g., a gate which only returns the current date and time).

   Category 3 gates are not tested because they have no effect on security.

Conclusion

PRIMOS satisfies the C2 Test Documentation requirement.

Design Documentation

Requirement

> Documentation shall be available that provides a description of the manufacturer's philosophy of protection and an explanation of how this philosophy is translated into the TCB. If the TCB is composed of distinct modules, the interfaces between these modules shall be described.

Applicable Features

To fulfill the design documentation requirement, Prime provided a function specification [2] of the design changes that were necessary for PRIMOS to meet the C2 Criteria requirements. This document covered these function changes at a limited level of detail.

To supplement the function specification, Prime provided highly detailed design documentation, which covered the major functional areas of PRIMOS. These documents, called Prime Engineering - Technical (PE-T) or Prime Engineering - Technical Internal (PE-TI) papers, contain valuable PRIMOS internal technical information in the form of functional specifications of a particular component of PRIMOS (written when the component was being designed) or simply internal PRIMOS documentation that is not available elsewhere in Prime documentation. A PE-TI paper is required as part of the design phase of any PRIMOS system component. The PE-T or PE-TI papers are indexed by subject and number.

Prime engineering has provided an informal security policy [8] which clearly provides a view of the major security mechanisms in PRIMOS. The informal security policy [8] provides a description of the ACL mechanism and the Ring 0 gated procedure protection mechanism. The informal security policy [8] does not include any discussion of hardware protection mechanisms. This document is currently a Prime Engineering - Technical Internal paper and is not available for general distribution. Copies may be obtained from Prime Engineering on request.

Prime also has a PRIMOS Internals Manual which may be acquired by taking the PRIMOS Internals class as offered by Prime Computer, Inc..

The only PRIMOS modules whose interfaces have been clearly defined are those which provide the interface to the TCB. The interface of the modules which are not a component of the TCB interface are not specifically documented in any manual, papers, or other formal documentation but are documented within the PRIMOS code as detailed comment headers for the given routine.

Conclusion

PRIMOS satisfies the C2 Design Documentation requirement.

FUNCTIONAL TESTING

The team tested PRIMOS with two major areas of concentration. The first phase of testing was aimed at providing the team with better assurance of the quality of Prime's test plan. To accomplish this the team tested those areas of PRIMOS , at the gate level, that the team felt had the best chance for penetration of the system.

The second phase of testing was aimed at testing PRIMOS's security features from the Command Processing Language (CPL) level. This testing was an attempt to exercise PRIMOS's security features and enhancements.

PHASE I: System Level Testing

The system level testing consisted of writing test programs to access and exercise the TCB gate interfaces. The following sections describe those tests.

The team testing took place in April of 1988 at the Prime Engineering facility in Framingham Massachusetts. Testing was performed on a 6350 running PRIMOS Revision 21.0.1.DODC2A.

Non-accessible Gates

The team tested the gates that are used by code not included in the evaluated system. In other words, a PRIMOS C2 site will have gates that do not function because they will either fail when accessed (the user type needed to execute the gate will NOT be configured into the C2 system) or will fail because they have none of the supporting library code necessary to perform their work. The following is a list of these types of gates:

- PRIMIX gates
- Network (SNA/X.25/RJE) gates
- IPC (remote/local) gates
- Node Status gates
- Network Terminal Server gates

The team wrote semi-automatic tests in PL/P to sample each type of gate. An error return was expected. All of the tests returned the expected error code thus showing that these gates are not accessible to users of the C2 system.

LOGIN SERVER System Call Level Test

The LOGIN SERVER validates users and in the process accesses the SAD. The team tested these gates to insure that a normal user cannot access these gates and thus get access to the LOGIN SERVER, or to passwords from the SAD.

The test was coded in PL/P and the call returned the error E$NRIT (INSUFFICIENT ACCESS RIGHTS) indicating that a normal user could not gain access to a gate that was reserved for the LOGIN SERVER process.

Category 2 Gate Code Review

The team reviewed the gates in category 2., as specified on page 65, "Test Documentation", in an attempt to find code paths that do not reference TCB security mechanisms when they should, or do not properly handle an error return from those calls. This testing is aimed at finding code that does not adhere to the criteria, specified by Prime in "Test Plan for C2 Security" [9], for these gates in category 2.

The team selected 13 code modules at random, representing 23 of the 97 total gates in category 2. The code modules were written in PL/P, Fortran and assembler. The modules were reasonably small and well structured making the code review relatively easy.

The team found no invalid code paths.

Ring 0 and Invalid Segment Pointers

The team attempted to gain access to ring 0 data structures by passing faulty pointers (e.g., ones that have bogus ring 3 addresses and ones that claim ring 0 access) to ring 0 entry points. This testing was aimed at circumventing the gate level protection and/or exploiting design flaws particular to ring 0 (e.g., NO ring 0 fault handler).

The test was coded in PL/P and consisted of a call to AC$LST (the ACL list system call) which requires at least one input parameter pointing to a buffer in the user program which would hold the ACL. That pointer value was manipulated using certain built-in PL/P pointer functions which allow a user to manipulate the various parts of a virtual address (ring #, segment #, offset).

The test always set the ring field to zero (0) and allowed the use of a different segment number by reading one in interactively. The use of a different segment number allowed the test to probe different sections of memory, which resulted in different types of program faults (e.g., ACCESS_VIOLATION and ILLEGAL_SEGNO).

Whenever PRIMOS is passed a pointer from ring 3, the ring bits are automatically maximized (set to 3). One part of this test only modified the pointer ring bits to zero (0) and displayed the pointer value before and after the AC$LST call. The pointer value did not change after the call and the call worked correctly (the ACL was read into the ring 3 buffer). Since PL/P passes parameters by reference (i.e., passes the address of the actual parameter rather than its value), the conclusion drawn was that the pointer passed was first copied, the ring was maximized, and this new pointer was used for the actual call. Thus, it can be concluded that PRIMOS copies the system call arguments from a ring 3 stack to a ring 0 stack and does not modify the input only parameters.

Random Gate Level Tests

The team performed exhaustive tests of a random sampling of gates with various types of parameters. The following tests were aimed at providing the team added assurance of the quality of Prime's test suite. To accomplish this aim, seven gates were chosen. The description and criteria for choosing these particular gates is as follows:

CREA$$: This gate has been superceded by a package of more specialized gates. The team wished to ensure that the system security was also applied to obsolete gates.

SEM$WT,
SEM$TN,
SEM$TS: These gates are the interfaces to the numbered semaphore mechanism. Since DAC was added to this mechanism late in the testing cycle, the team wanted to ensure that the added DAC enhancements worked as specified.

SPAWN$,
FORK$,
PHNTM$: These are used to create children processes. The team wanted to ensure that no undeserved privileges can be created by providing these calls with invalid parameters.

The test code for the seven gates was written in PL/P and were coded such that they could be fit into Prime's automated test suite.

All of the tests were aimed at testing the specific error returns, as specified in the "Subroutine Reference Guide" [7], and to determine whether the gates performed as expected. The error returns that were tested and their descriptions are listed below:

E$EXST:	The specified object already exists.
E$NRIT:	The caller has no rights to perform the given function.
E$BNAM:	The name provided by the caller is invalid.
E$BPAR:	The parameter passed in not of the expected type.

During the testing of the seven gates, the team found no unexpected results (i.e., all invalid information passed to the gate was intercepted and the appropriate error code was returned). The team did not find any obvious ways to subvert the tested gates.

Object Reuse

The team tested PRIMOS to make sure its object reuse mechanisms cannot be circumvented. The process registers do not pose a object reuse problem as described on (see page 58, "Object Reuse") and thus this area was not tested.

Memory reuse was tested by examining a memory map of the storage allocated to the tester's process. A memory allocation request was performed so that new unused memory would be allocated. The contents of this allocated area of memory were then examined. From this it was determined that memory that had been previously allocated and used by the process, is not zeroed on allocation. However, newly allocated memory is zeroed when it is first allocated to the process.

File space reuse was tested by creating a new file, allocating file space and attempting to examine the contents. In the process of this testing, it was determined that file space is allocated sequentially (it is not possible to allocate logically non-contiguous space beyond the current physical end-of-file). In addition, PRIMOS insures that an area of the file cannot be read before it has been written. Attempts to allocate space beyond the end-of-file or to reference an area of the file that had not been written were greeted with an error on the request.

PHASE II: CPL Level Testing

CPL is the Prime's command language, that represents the user's primary interface to the system. This testing consisted of manual tests to exercise various security related CPL interfaces to the system. All of these tests were documented in COMO runs. The following sections describe these tests.

DAC Tests

The Team's DAC tests were manual tests designed to test the functionality of PRIMOS's DAC system. The tests consisted of setting a variety of access rights to objects in the system and trying to access these objects with, and without, the correct access rights. All tests produced the expected outcome.

LOGIN SERVER CPL Level Test

The team created tests to exercise PRIMOS's I&A mechanisms.

The following EDIT_PROFILE directives were manipulated in order to test the password capabilities of PRIMOS.

- FORCE_PASSWORD [-ON/-OFF]
- NO_NULL_PASSWORD [-ON/-OFF]
- MINIMUM_PASSWORD_LENGTH [length]
- CHANGE_USER [username] -PASSWORD

FORCE_PASSWORD -OFF

If a user had a null password, the LOGIN SERVER would not even ask for a password and just do an automatic log in. A user with a non-null password is still asked for the password.

FORCE_PASSWORD -ON

If a user had a null password, the LOGIN SERVER would prompt for a password, and user types in a ARRIAGE_RETURN to log in. A user with a non-null password is always asked for their password.

NO_NULL_PASSWORD -ON

Any users added after that command would have to have a non-null password of a length determined by the MINIMUM_PASSWORD_LENGTH directive. The system would display which current users had null passwords.

MINIMUM_PASSWORD_LENGTH [LENGTH]

This directive allows the system administrator to set the minimum password length for users who will be added in the future. The SAD UPD entries do not store the passwords in ASCII format or save the length of the ASCII password. Thus, PRIMOS cannot know which users currently have a password which does not adhere to this minimum length. EDIT_PROFILE will however list users who currently have null passwords. Setting this value does not force log outs of users whose accounts do not adhere to the set value.

The value set for the MINIMUM_PASSWORD_LENGTH directive overrides the NO_NULL_PASSWORD -OFF directive. Adding a user after setting a minimum length for a password forces that user to have that length password and a null password cannot be given to that user.

CHANGE_USER [USERNAME] -PASSWORD

This EDIT_PROFILE directive allows the system administrator or project administrator to change the password of a user. This does not force an immediate log out of the user but requires the user to provide the correct password on the next log in attempt. The password requirements adhere to the NO_NULL_PASSWORD and MINIMUM_PASSWORD_LENGTH directives as currently set.

CHANGE_PASSWORD

Any user may enter this command. The command format has two variations.

1. CHANGE_PASSWORD [old_password], or

2. CHANGE_PASSWORD -PROMPT

The first one requires the user to specify their old password on the [illegible]and line, and so the password is echoed on the terminal. The second version requests a p[illegible] the old password.

When required to type in their old password, it is not echoed. The user is then asked for a new password that does not match the old password and conforms to the NO_NULL_PASSWORD and MINIMUM_PASSWORD_LENGTH directives.

If the old password is incorrectly specified, the user is given an error, and the old password remains in effect.

It was also noted that whenever users log in, they are informed as to whether anyone else is currently logged in under the same user name.

Testing of the Audit Mechanism

The team created an "audit file full" condition to see how PRIMOS manages it.

The test itself was written in CPL, a PRIMOS command level language, and required some administrative set up. The administrative set up involved starting the audit collection utility with its log file to be written on a small disk partition and monitoring all file system events. This was done so that filling up the log file could be accomplished very quickly and because the test would be generating file system auditable events. The test itself involved repeatedly accessing a file to which it had no access. Once the audit file filled up, the system is supposed to suspend all user processes and only allow a limited number of commands to be executed at the system console. This was exactly what happened; all work was suspended until the proper actions were taken at the system console to remedy the problem.

PRIMOS Indirection Loop Test

The purpose of this test was to determine whether an indirection loop would hang the machine.

When the test program was executed, the microcode generated a RESTRICTED INSTRUCTION program fault very quickly. It was not possible to determine the number of loops it took, but the machine did not hang.

EVALUATORS' COMMENTS

This section consists of the comments and opinions the evaluation team developed during use of and testing of PRIMOS.

Informal Security Policy Model

Although there is no requirement for any type of security policy model at the C2 evaluation class, Prime provided the evaluation team with an informal model for PRIMOS. This model describes the subjects and objects, their interactions, and access control mechanisms within PRIMOS. The general team consensus is that this document provides a useful introduction to the security features of PRIMOS.

TCB Interface Control

The team had expressed concerns about the apparent lack of control over the addition of interfaces to the PRIMOS TCB. From our initial study of PRIMOS, it was our opinion that Prime's configuration management in this area was somewhat lacking. Prime addressed this issue by providing a mechanism to limit the addition of interfaces to the TCB. This mechanism provides an acceptable means of limiting the adding of additional interfaces, by forcing these additions to go through a more stringent review process by an designate entity within Prime. The team feels that the current mechanism for TCB interface control will be very beneficial to Prime in its overall configuration management of PRIMOS.

Robustness of the Audit Mechanism

In satisfying the C2 audit requirements, Prime built what the team feels is a high quality mechanism for the gathering and viewing of audit data. These mechanisms have already been described in the system overview section of the report (see page 49, "Audit"), so they will not be redescribed here. Of particular interest to the team, was the extent that Prime went to to guarantee that no audit data could be lost. This fact is illustrated by the way in which PRIMOS handles/prevents the overflowing of the audit log, and the fact that Prime provides a tool to extract audit records from a system image after a system crash.

Awkwardness of Various Aspects of the DAC Mechanisms

Although PRIMOS provides a good DAC implementation, the mechanism for providing default protection is considered by the team to be [illegible]. Default protection for files is determined from the protection of the current (or nearest) ACL protected directory. Since the protection flags on a directory do not directly apply to the protection afforded a file, there is a mapping scheme that some end users might find awkward. Although this mechanism does not inherently harm system security, it is the team's opinion that this mechanism could provide an area of difficulty in understanding for the end users.

PRIMOS provides a DAC mechanism to limit access to devices. Although this mechanism currently meets the TCSEC C2 DAC requirements, the team feels that Prime's chosen implementation provides a greater likelyhood for mistakes to be made in passing out access to the media for a given device. The mechanism depends on the interpretation of the DAC on the given medium to be performed by the operator; since the interpretation is being performed by a human, it is the team's opinion that the likelyhood for this interpretation to be performed incorrectly (human error), is greater.

EVALUATED HARDWARE COMPONENTS

List of Evaluated Components

The hardware covered by this evaluation includes all processors in the Prime 50 Series product line. The primary requirement for hardware evaluation is that the hardware function properly. This is verified by the system integrity tests (see page 62, "System Integrity") and was not given a detailed evaluation by the team. The integrity assurances provided by the Prime-supplied diagnostic tests are satisfactory.

Standard Systems by CPU Type

This section lists, by system type, the Prime identification numbers and description, of all hardware components that are covered by this evaluation. To operate in correspondence with the C2 rating, the hardware configuration of an installation must contain only components listed in this section.

2350-011	Standard 2350 System	2MB ECC MOS Memory (1 BD), 60MB Disk, 60MB Cartridge Tape Unit, Disk/Tape Controller, 8 Async Lines, 1 Synch Line (ICS1), PT200 System Console, modem, and executable PRIMOS.
2350-021	Standard 2350 System	4MB ECC MOS Memory (1 BD), 120MB Disk, 60MB Cartridge Tape Unit, Disk/Tape Controller, 16 Async Lines, PT200 System Console, modem, and executable PRIMOS.
2350-030	Standard 2350 System	4MB ECC MOS Memory (1 BD), (60HZ) 120MB Disk, 60MB Cartridge Tape Unit, Disk/Tape Controller, ICS3 with 4 Async Lines, modem, and executable PRIMOS.
2350-031	Standard 2350 System	4MB ECC MOS Memory (1 BD), (60HZ) 120MB Disk, 60MB Cartridge Tape Unit, Disk/Tape Controller, ICS3 with 4 Async Lines, PT200 system console, modem, and executable PRIMOS.
2350-041	Standard 2350 System	4MB ECC MOS Memory (1 BD), (60HZ) 258MB Disk, 60MB Cartridge Tape Unit, Disk/Tape

		Controller, ICS3 with 4 Async Lines, PT200 system console, modem, and executable PRIMOS.
2450-021	Standard 2450 System	4MB ECC MOS Memory (1 BD), 120MB Disk, 60MB Cartridge Tape Unit, Disk/Tape Controller, 16 Async Lines, PT200 system console, modem, and executable PRIMOS.
2450-030	Standard 2450 System	4MB ECC MOS Memory (1 BD), (60HZ) 120MB Disk, 60MB Cartridge Tape Unit, Disk/Tape Controller, ICS3 with 4 Async Lines, modem, and executable PRIMOS.
2450-031	Standard 2450 System	4MB ECC MOS Memory (1 BD), (60HZ) 120MB Disk, 60MB Cartridge Tape Unit, Disk/Tape Controller, ICS3 with 4 Async Lines, PT200 system console, modem, and executable PRIMOS.
2450-041	Standard 2450 System	4MB ECC MOS Memory (1 BD), (60HZ) 258MB Disk, 60MB Cartridge Tape Unit, Disk/Tape Controller, ICS3 with 4 Async Lines, PT200 system console, modem, and executable PRIMOS.
2655-ELA	Standard 2655 System	4MB ECC MOS Memory (1 BD), 496MB FMD with Intelligent Controller, and office peripheral cabinet.
2655-ELC	Standard 2655 System	4MB ECC MOS Memory (1 BD), 496MB FMD with Intelligent Controller, office peripheral cabinet, and system console.
2655-FLA	Standard 2655 System	4MB ECC MOS Memory (1 BD), 496MB FMD with Intelligent Controller, Streaming Mag Tape Subsystem, and office peripheral cabinet.

2655-FLC	Standard 2655 System	4MB ECC MOS Memory (1 BD), 496MB FMD with Intelligent Controller, Streaming Mag Tape Subsystem, office peripheral cabinet, and system console.
2655-JLA	Standard 2655 System	8MB ECC MOS Memory (2 BDS), two 496MB FMDs with Intelligent Controller, Streaming Mag Tape Subsystem, ICS3 with 16 Async Lines (4 CLAC204S), and two office peripheral cabinets.
2655-JLC	Standard 2655 System	8MB ECC MOS Memory (2 BDS), two 496MB FMDs with Intelligent Controller, Streaming Mag Tape Subsystem, ICS3 with 16 Async Lines (4 CLAC204S), two office peripheral cabinets, and system console.
2755-ALC	Standard 2755 System	4MB ECC MOS Memory (1 BD), 496MB FMD with Intelligent Controller, office peripheral cabinet, system console, and executable PRIMOS.
2755-BLA	Standard 2755 System	4MB ECC MOS Memory (1 BD), 496MB FMD with Intelligent Controller, Streaming Mag Tape Subsystem, office peripheral cabinet, and executable PRIMOS.
2755-BLC	Standard 2755 System	4MB ECC MOS Memory (1 BD), 496MB FMD with Intelligent Controller, Streaming Mag Tape Subsystem, office peripheral cabinet, system console and executable PRIMOS.
2755-CLA	Standard 2755 System	8MB ECC MOS Memory (2 BDS), two 496MB FMDs with Intelligent Controller, Streaming Mag Tape Subsystem, ICS3 with 16 Async Lines (4 CLAC204), two office peripheral cabinets, and executable PRIMOS.
2755-CLC	Standard 2755 System	8MB ECC MOS Memory (2 BDS), two 496MB FMDs with Intelligent Controller, Streaming Mag

		Tape Subsystem, ICS3 with 16 Async Lines (4 CLAC204), two office peripheral cabinets, system console, and executable PRIMOS.
6350-ASA	Standard 6350 CPU (60HZ)	32MB ECC MOS Memory (4 BDS), 770MB FMD with Intelligent Controller, peripheral cabinet, and executable PRIMOS.
6350-ASA-A	Standard 6350 CPU (50 HZ)	32MB ECC MOS Memory (4 BDS), 770MB FMD with Intelligent Controller, peripheral cabinet, and executable PRIMOS.
6350-ASC	Standard 6350 CPU (60 HZ)	32MB ECC MOS Memory (4 BDS), 770MB FMD with Intelligent Controller, peripheral cabinet, PT200 color system console, and executable PRIMOS.
6350-ASC-A	Standard 6350 CPU (50 HZ)	32MB ECC MOS Memory (4 BDS), 770MB FMD with Intelligent Controller, peripheral cabinet, and executable PRIMOS.
6350-CSA	Standard 6350 CPU (60 HZ)	32MB ECC MOS Memory (4 BDS), three 770MB FMDs with Intelligent Controller, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, and executable PRIMOS.
6350-CSA	Standard 6350 CPU (50 HZ)	32MB ECC MOS Memory (4 BDS), three 770MB FMDs with Intelligent Controller, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, and executable PRIMOS.
6350-CSC	Standard 6350 CPU (60 HZ)	32MB ECC MOS Memory (4 BDS), three 770MB FMDs with Intelligent Controller, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, PT200 color system console, and executable PRIMOS.

6350-CSC-A	Standard 6350 CPU (50 HZ)	32MB ECC MOS Memory (4 BDS), three 770MB FMDs with Intelligent Controller, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, PT200 color system console. and executable PRIMOS.
6350-ESA	Standard 6350 CPU (60 HZ)	32MB ECC MOS Memory (4 BDS), five 770MB FMDs with two Intelligent Controllers, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, and executable PRIMOS.
6350-ESA-A	Standard 6350 CPU (50 HZ)	32MB ECC MOS Memory (4 BDS), five 770MB FMDs with two Intelligent Controllers, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, and executable PRIMOS.
6350-ESC	Standard 6350 CPU (60 HZ)	32MB ECC MOS Memory (4 BDS), five 770MB FMDs with two Intelligent Controllers, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, PT200 color system console, and executable PRIMOS.
6350-ESC-A	Standard 6350 CPU (50 HZ)	32MB ECC MOS Memory (4 BDS), five 770MB FMDs with two Intelligent Controllers, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, PT200 color system console, and executable PRIMOS.
6550-ASA	Standard 6550 CPU (60 HZ)	32MB ECC MOS Memory (4 BDS), 770MB FMD with Intelligent Controller, peripheral cabinet, and executable PRIMOS.
6550-ASA-A	Standard 6550 CPU (50 HZ)	32MB ECC MOS Memory (4 BDS), 770MB FMD with Intelligent Controller, peripheral cabinet, and executable PRIMOS.

6550-ASC	Standard 6550 CPU (60 HZ)	32MB ECC MOS Memory (4 BDS), 770MB FMD with Intelligent Controller, peripheral cabinet, PT200 color system console, and executable PRIMOS.
6550-ASC-A	Standard 6550 CPU (50 HZ)	32MB ECC MOS Memory (4 BDS), 770MB FMD with Intelligent Controller, peripheral cabinet, PT200 color system console, and executable PRIMOS.
6550-CSA	Standard 6550 CPU (60 HZ)	32MB ECC MOS Memory (4 BDS), three 770MB FMDs with Intelligent Controller, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, and executable PRIMOS.
6550-CSA-A	Standard 6550 CPU (50 HZ)	32MB ECC MOS Memory (4 BDS), three 770MB FMDs with Intelligent Controller, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, and executable PRIMOS.
6550-CSC	Standard 6550 CPU (60 HZ)	32MB ECC MOS Memory (4 BDS), three 770MB FMDs with Intelligent Controller, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, PT200 color system console, and executable PRIMOS.
6550-CSC	Standard 6550 CPU (50 HZ)	32MB ECC MOS Memory (4 BDS), three 770MB FMDs with Intelligent Controller, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, PT200 color system console, and executable PRIMOS.

6550-ESA	Standard 6550 CPU (60 HZ)	32MB ECC MOS Memory (4 BDS), five 770MB FMDs with two Intelligent Controllers, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, and executable PRIMOS.
6550-ESA-A	Standard 6550 CPU (50 HZ)	32MB ECC MOS Memory (4 BDS), five 770MB FMDs with two Intelligent Controllers, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, and executable PRIMOS.
6550-ESC	Standard 6550 CPU (60 HZ)	32MB ECC MOS Memory (4 BDS), five 770MB FMDs with two Intelligent Controllers, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, PT200 color system console, and executable PRIMOS.
6550-ESC-A	Standard 6550 CPU (50 HZ)	32MB ECC MOS Memory (4 BDS), five 770MB FMDs with two Intelligent Controllers, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 in System Cabinet with 32 Async Lines, two peripheral cabinets, PT200 color system console, and executable PRIMOS.
9655-CSC	Standard 9655 System	4MB ECC MOS Memory (1 BD), 496MB FMD with Intelligent Controller, Streaming Mag Tape Subsystem, peripheral cabinet, and system console.
9655-GSC	Standard 9655 System	4MB ECC MOS Memory (1 BD), 496MB FMD with Intelligent Controller, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, peripheral cabinet, and system console.

9655-HSA	Standard 9655 System	8MB ECC MOS Memory (2 BDS), two 496MB FMDs with Intelligent Controller, ICS3 with 32 Async Lines (8 CLAC304S), and peripheral cabinet.
9655-HSC	Standard 9655 System	8MB ECC MOS Memory (2 BDS), two 496MB FMDs with Intelligent Controller, ICS3 with 32 Async Lines (8 CLAC304S), peripheral cabinet, and system console.
9655-MSC	Standard 9655 System	4MB ECC MOS Memory (1 BD), 300MB SMD and CRT console.
9655-VNC	Standard 9655 System	4MB ECC MOS Memory (1 BD), 496MB FMD with Intelligent Controller, peripheral cabinet, and system console.
9750-APF	Standard 9750 System	4MB ECC MOS Memory (2 BDS), two 315MB FMDs with one Controller, Streaming Magnetic Tape Subsystem, peripheral cabinet, and CRT console.
9750-CPF	Standard 9750 System	4MB ECC MOS Memory (2 BDS), three 315MB FMDs with one controller, peripheral cabinet, and CRT console.
9750-DPF	Standard 9750 System	4MB ECC MOS Memory (2 BDS), two 315MB FMDs with one Controller, ICS Model II with 32 Asynch Lines, peripheral cabinet, and CRT console.
9750-EPF	Standard 9750 System	4MB ECC MOS Memory (2 BDS), 315MB FMD, Streaming Mag Tape Subsystem, ICS Model II with 32 Async Lines, peripheral cabinet, and CRT consoie.

9750-FPF	Standard 9750 System	4MB ECC MOS Memory (2 BDS), 315MB FMD, 800/1600/6250 BPI 50 IPS GCR Tape, and CRT console.
9750-MNF	Standard 9750 System	4MB ECC MOS Memory (2 BDS), 300MB SMD, and CRT console.
9750-PNF	Standard 9750 System	4MB ECC MOS Memory (2 BDS), 675MB FMD, and CRT console.
9750-TNF	Standard 9750 System	4MB ECC MOS Memory (2 BDS), 315MB FMD, peripheral cabinet, and CRT console.
9755-CSC	Standard 9755 System	8MB ECC MOS Memory (2 BDS), 496MB FMD with Intelligent Controller, Streaming Mag Tape Subsystem, peripheral cabinet, and CRT console.
9755-FSC	Standard 9755 System	8MB ECC MOS Memory (2 BDS), 496MB FMD with Intelligent Controller, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, peripheral cabinet, and color system console.
9755-GSA	Standard 9755 System	8MB ECC MOS Memory (2 BDS), 496MB FMD with Intelligent Controller, ICS3 with 32 Async Lines (8 CLAC304S), and peripheral cabinet.
9755-GSC	Standard 9755 System	8MB ECC MOS Memory (2 BDS), 496MB FMD with Intelligent Controller, ICS3 with 32 Async Lines (8 CLAC304S), peripheral cabinet, and color system console.
9755-KSC	Standard 9755 System	8MB ECC MOS Memory (2 BDS), 773MB FMD with Intelligent Controller, 800/1600/6250 BPI 50 IPS Mag Tape Subsystem, ICS3 with 32 Async Lines, two peripheral cabinets, and color system console.

9755-MNC	Standard 9755 System	8MB EC C MOS Memory (2 BDS), 30 OMB SMD, and color system console.
9755-VNC	Standard 9755 System	8MB ECC MOS Memory (2 BDS), 496MB FMD with Intelligent Controller, peripheral cabinet, and color system console.
9755-WNC	Standard 9755 System	8MB ECC MOS Memory (2 BDS), 773MB FMD with Intelligent Controller, peripheral cabinet, color system console and executable PRIMOS.
9955-APF	Standard 9955 System	8MB ECC MOS Memory (1 BD), two 315MB FMDs with controller, Streaming Mag Tape Subsystem, peripheral cabinet, and CRT console.
9955-BPF	Standard 9955 System	8MB ECC MOS Memory (1 BD), 315MB FMD, 800/1600/6250 BPI 75 IPS Mag Tape Subsystem, peripheral cabinet, and CRT console.
9955-CPF	Standard 9955 System	8MB ECC MOS Memory (1 BD), three 315MB FMDs with controller, peripheral cabinet, and CRT console.
9955-DPF	Standard 9955 System	8MB ECC MOS Memory (1 BD), two 315MB FMDs, ICS Model II with 32 Async Lines, peripheral cabinet, and CRT console.
9955-EPF	Standard 9955 System	8MB ECC MOS Memory (1 BD), 315MB FMD, Streaming Mag Tape Subsystem, ICS Model II with 32 Async Lines, peripheral cabinet, and CRT console.
9955-FPF	Standard 9955 System	8MB ECC MOS Memory (1 BD), 315MB FMD, 800/1600/6250 BPI 50 IPS GCR Mag Tape Subsystem, and CRT console.

9955-GPF	Standard 9955 System	8MB ECC MOS Memory (1 BD), two 675MB FMDs with two controllers, 800/1600/6250 BPI 50 IPS GCR Mag Tape Subsystem, and CRT console.
9955-HPF	Standard 9955 System	8MB ECC MOS Memory (1 BD), three 315MB FMDs withtwo controllers, 800/1600/6250 BPI 50 IPS GCR Mag Tape Subsystem, and CRT console.
9955-JPF	Standard 9955 System	16MB ECC MOS Memory (2 BD), two 675MB FMDs with two controllers, 800/1600/6250 BPI 50 IPS GCR Mag Tape Subsystem, and CRT console.
9955-LPF	Standard 9955 System	16MB ECC MOS Memory (2 BD), two 675MB FMDs with four controllers, 800/1600/6250 BPI 50 IPS GCR Mag Tape Subsystem, and CRT console.
9955-MNF	Standard 9955 System	8MB ECC MOS Memory (1 BD), 30 OMB SMD, CRT Console.
9955-PNF	Standard 9955 System	8MB ECC MOS Memory (1 BD), 675MB FMD, CRT Console.
9955-TNF	Standard 9955 System	8MB ECC MOS Memory (1 BD), 315MB FMD, 1 Peripheral cabinet, CRT Console.
9955M2-16MB	Standard 9955 II CPU	16MB of ECC MOS Memory (2 BD), CRT Console.
9955M2-8MB	Standard 955 II CPU	8MB of ECC MOS Memory (1 BD), CRT Console.
955M2-ESC	Standard 9955 II System	16MB ECC MOS Memory (2 BD), 1 496MB FMD with controller, 800/1600/6520 BPI 50 IPS Tape subsystem, 1 peripheral cabinet, color system console.

9955M2-GTC	Standard 9955 II System	16MB ECC MOS Memory (2 BD), 3 496MB FMD with controller, 800/1600/6520 BPI 50 IPS Tape subsystem, 2 peripheral cabinets, color system console, executable PRIMOS.
9955M2-HTC	Standard 9955 II System	16MB ECC MOS Memory (2 BD), 3 773MB FMDS with controller, 800/1600/6520 BPI 50 IPS Tape subsystem, ICS3 with 32 ASYNC lines, 2 peripheral cabinets, color system console, executable PRIMOS.
9955M2-LTC	Standard 9955 II System	32MB ECC MOS Memory (4 BD), 5 773MB FMDS with 2 controllers, 800/1600/6520 BPI 50 IPS Tape subsystem, ICS3 with 32 ASYNC lines, 2 peripheral cabinets, color system console, executable PRIMOS.
9955M2-MNC	Standard 9955 II System	16MB ECC MOS Memory (2 BD), 300MB SMD, color system console.
9955M2-VNC	Standard 9955 II System	16MB ECC MOS Memory (2 BD), 496MB FMD with controller, 1 peripheral cabinet, color system console.
9955M2-WMC	Standard 9955 II System	16MB ECC MOS Memory (2 BD), 773MB Disk with controller, 1 peripheral cabinet, color system console, executable PRIMOS.

EVALUATED SOFTWARE COMPONENTS

The software covered by this evaluation includes PRIMOS Revision 21.0.1.DODC2A with all of its bundled support software. A separately priced audit facility (see page 49, "Audit"), is also required for C2 configurations. This package is included in the C2 release.

List of Evaluated Components

This section lists, those software products required in a C2 configured site. When ordering PRIMOS Revision 21.0.1.DODC2A from Prime, the orderer must specify two product numbers. The first number pertains to the evaluated revision of PRIMOS. The second number pertains to those software elements of PRIMOS that are necessary for a C2 configured site (e.g., the auditing facility, C2 configuration file). The second number differs for each target processor. This difference does not reflect any software differences in PRIMOS; it is merely a marketing tool for pricing PRIMOS differently based on the processor type.

PRIMOS Revision 21.0.1.DoD_C2a

All of PRIMOS and its basic support software is bundled together in one package with one part number. This basic support software includes command software, standard library code, help files, on line documentation, runfiles, DSM, SPOOL, BATCH, and other basic support software. The following is the part number used to order the evaluated version of PRIMOS.

850300	Prime Computer, Inc. with basic C2 support

PRIMOS Revision 21.0.1.DoD_C2a Complete C2 Support Package

The complete C2 support packages includes the auditing facilities, a C2 configuration file, the correct ACLing, as specified in the Trusted Facility Manual [5], of system specific file system objects, and the removal of those software components disallowed for a C2 site (e.g., networking support) from the standard distribution.

850400C-	P1 For CPUs 2350, 2450
850400C-	P2 For CPUs 2250, 2550, 2655, 2755
850400C-	P4 For CPUs numbered 9x5x
850400C-	P5 For CPUs 6350

850400C-	P6 For CPUs 6550
850400C-	P1S Upgrade for CPUs 2350, 2450
850400C-	P2S Upgrade for CPUs 2250, 2550, 2655, 2755
850400C-	P4S Upgrade for CPUs numbered 9x5x
850400C-	P5S Upgrade for CPUs 6350
850400C-	P6S Upgrade for CPUs 6550

References

[1] Dijkstra E. W., "Cooperating Sequential Processes" in Programming Languages, ed. F. Genuys, Academic Press, 1968.

[2] C2 Functional Specification 2.0, Prime Computer, Inc. September 22, 1986.

[3] Prime Engineering Technical Report 1122, Prime Computer, Inc., Revision 11.0, January 29, 1987.

[4] Security Features User's Guide, Prime Computer, Inc., First Edition for Revision 21.0, May 1987.

[5] System Administrator's Guide, Vol. III, Prime Computer, Inc., First Edition for Revision 21.0, May 1987.

[6] System Architecture Reference Guide, Prime Computer, Inc., Updated for Revision 20.1, May 1986.

[7] Subroutine Reference Guide, Prime Computer, Inc., Updated for Revision 20.1, May 1986.

[8] PRIMOS's Security Policy, Prime Computer, Inc., December 1986.

[9] Test Plan for C2 Security , Prime Computer, Inc., September 1987.

ACRONYMS

ACL	Access Control List
ALU	Arithmetic Logic Unit
AMLC	Asynchronous Multiplexor Line Controller
CPL	Command Programming Language
CSU	Control Store Unit
DAM	Direct Access Method"
DLL	Down Line Load
DMA	Direct Memory Access
DMC	Direct Memory Command
DMQ	Direct Memory Queue
DTAR	Descriptor Table Address Register
DSM	Distributed System Management
ECB	Entry Control Block
ECL	Emitter Coupled Logic
EPF	Executable Program File
IAP	Initial Attach Point
IGUANA	Asynchronous terminal controller
IPC	Inter-Process Communications
IPU	Instruction Preprocessor Unit

ISC	Inter Server Communications
ISU	Instruction Stream Unit
LSR	Login Server
MDLC	Another type of line controller
MFD	Master File Directory
MIK	Machine Independent Kernel
NPX	Network Process Exchange
NSS	Network Status Server
PA	Project Administrator
PCB	Process Control Block
PEU	Processor Execution Unit
PIT	Process Interval Timer
PMT	Page Map Table
PRIMIX	Unix interface to PRIMOS
PRATIO	Having to do with paging to multiple devices
PXM	Process Exchange Mechanism
QAMLC	Queued Asynchronous Multiplexor Line Controller
R0AM	Ring 0 Access Method
RESUS	Remote System User (part of DSM)
RF	Register File

RIPC	Remote Inter-Process Communications (IPCs)
SA	System Administrator
SAD	System Administrator Directory
SAM	Sequential Access Method
SDT	Segment Descriptor Table
SDW	Segment Descriptor Word
SIM	System Information and Metering (part of DSM)
SMLC	Synchronous Multiplexor Line Controller
SNA	System Network Architecture (IBM interface)
SSU	Stream Synchronization Unit
STLB	Segmentation Table Lookaside Buffer
UFD	User File Directory
UMH	Unsolicited Message Handler (part of DSM)
UPS	Uninterruptible Power Supply

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution Unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CSC-EPL-88/009	5. MONITORING ORGANIZATION REPORT NUMBER(S) S231,362
6a. NAME OF PERFORMING ORGANIZATION National Computer Security Center 6b. OFFICE SYMBOL (If applicable) C12	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Ft. George G. Meade, MD 20755-6000	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT NO.

11. TITLE (Include Security Classification)
PRIME COMPUTER, INC. PRIMOS 21.0.1.DODC2A

12. PERSONAL AUTHOR(S)
Newton, Richard ; Summers, David ; Wolfe, Harold; Coplin, Myron

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr, Mo., Day)	15. PAGE COUNT
Final	FROM TO	880718	108

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
NCSC ACLs

19. ABSTRACT (Continue on reverse side if necessary and identify by block number)
The security protection provided by the Prime Computer, Inc. PRIMOS Revision 21.0.1.DODC2A operating system has been evaluated by the National Computer Security Center (NCSC). The NCSC evaluation team has determined that PRIMOS satisfies all the specified requirements of class C2 of the Department of Defense Trusted Computer System Evaluation Criteria, when configured as a stand-alone system according to the most secure manner as described in the Trusted Facility Manual and while running on Prime 50 Series Hardware.

This report documents the findings of the evaluation.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL DENNIS E. SIRBAUGH	22b TELEPHONE NUMBER (Include Area Code) (301)859-4458	8b. OFFICE SYMBOL C12

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE